Report 3: Restaurant Automation

GROUP #1

FOOD BYTE

Team Members:

Akshay Gangal

Anvitha Selva Kumar

Ashwin Channakeshava

Prince Bose

Priyanka Rajan

Sarthak Vengurlekar

Suraj Nagaraj

Yashasvi Khatavkar

“All the members in our team contributed equally!”

SUMMARY OF CHANGES:

From Report 1 to Report 2, we modified the use cases, use case diagrams and description. We added an extra use case for Data set creation using mathematical model and another use case for Clustering. We also added explanation for the mathematical model and how it is applicable to our system. But we haven't made any design changes from Report 2 to Report 3.

TABLE OF CONTENTS

1. PROJECT DESCRIPTION

Using our restaurant automation application, restaurants can increase the efficiency of daily operations with a central system to track all aspects of restaurant management, individual customer transactions, observing profit, costs, revenue, analyzing and optimizing the menu based on collected data.

Lately, data analytics is being used in many fields to derive insight on how the business/organization is running as it enables them to look for meaningful patterns and correlations in their business. Companies benefit from this as they are entitled to make better decisions in the future by analyzing data from the past. Our system incorporates the concepts of data analytics in the restaurant business to enable features such as trend analysis based on geography of the restaurant, customer density analysis based on time of day, item-based revenue maximization and inventory management.

2. CUSTOMER STATEMENT OF REQUIREMENTS

For this project, the customer will be the restaurant management.

2.1. Inventory Management to Reduce Waste

2.1.1. Problem Diagnosis

Managing a restaurant is a very difficult job. There are many variables surrounding all elements. Starting a restaurant is not an inexpensive idea. To keep the expense according to the budget, many factors are to be considered. One of the main problems that almost all restaurants face is to determine the amount of food to be bought in. This problem is steeper for newly commenced restaurants. If a good deal of food is bought, and the demand is less, the food may be wasted, and it may contaminate the newer products. On the other hand, if the stocked food is not enough i.e. if the demand is more, the restaurant may run out of food quickly. This may cause the restaurant to close before the allotted closing time and may leave some customers displeased. Each ingredient has a different usage rate, so the demand of each ingredient changes based on the usage. If the old ingredients are not disposed and the newer products are placed near the older products, this may result in the contamination of the newer ingredients.

2.1.2 Proposed Solution

Having a detailed Inventory, helps us to determine the amount of food that we have and the amount of food that we need. It helps us avoid surplus as well as food shortages. Based on the data and the trend, we will try to predict the rate of use of certain ingredients and will generate the adequate quotation of the required ingredients before they replenish. This will be done by using Data Analytics, analyzing the trend of the consumption of each ingredient over time, and thus, notifying the pantry of the requirements well in advance.

The average waste rate for full-service eateries is 3.11% [1]. Hence, we will also focus on the First In First Out technique to ensure that no food waits for a long time on the shelf before usage. This will help the restaurant avoid wastage of food that has been sitting the shelf for a long time.

2.1.3. Plan of Work and Product Ownership

Our inventory will consist of the following information:

1. Item Name
2. Unit of Measurement
3. Inventory Amount
4. Unit Price
5. Total Cost
6. Usage in Dishes

Prince and Sarthak will use the data along with: usage and quantity of each ingredient in a dish, the frequency of the number of tables that ordered for that dish and average consumption of the dish per week/month, that will help us to understand the rate of consumption of an ingredient.

We will also suggest changes in the menu, if an ingredient has been waiting on the shelf for a long time and needs to be used to avoid contamination.

Prince and Sarthak: Our plan over the next few weeks is to make and link Ingredient Lookup tables and Ingredient Availability tables to our Server side program. Once we do that, we plan to read that data and implement different Data Analytic techniques to infer conclusions and possible solutions.

We want to add the following functionalities:

- Ingredient Consumption: To understand the rate of usage of specific ingredients, maximize profitability, and reduce wastage.
- Suggest Changes to the Menu: To increase profitability by making dishes that will help replenish the least used ingredients and avoid wastage.

2.2 Trend Analysis Based on Geography

2.2.1. Problem diagnosis

Loss occurring because of using same menu across different geographical locations

The location of a restaurant has a significant impact on the sales of food items. Restaurants normally tend to create a fixed menu irrespective of the choice of people living in that area. For example, people living in the west coast prefer fusion cooking in contrast to the people in east coast who prefer classics. In this case, a restaurant can incur a loss if it does not customize its menu according to the food preference of the people living in that area. In addition to this they do not generally discard items with no or fewer sales.

2.2.2. Proposed Solution

To regulate the price of food items based on the sales at specific locations. The database will keep a track of the sales information and the server will have a popularity index. This data will be fetched by the data analytics module which will utilize an algorithm that will inform the server about a range of prices that can be changed for this product and a notification to the inventory to increase the stock.

Tracking of unsold items on the menu. If an item on the menu is not being sold as expected, we will either reduce its price, advertise with an offer or remove it from the menu. In this case we can use the same popularity index for regulation of the unpopular items.

2.2.3. Plan of Work

Yashasvi and Akshay: Our plan to accomplish in the next few weeks will be to create sample data sets of different food items of a restaurant at different locations and to categorize them into different sub groups according to their sales and apply analytics to this data to increase the profits by customizing the menu. Following are the groups based on popularity index:

1) Favorite

2) Least sold

3) Can be improved

2.3. Analysis Based on Customer Density

2.3.1. Problem Diagnosis

In the current day working scenario of restaurants, the management does not tend to concentrate more on the number of customers visiting their restaurants during different periods of the day. The hotel management always expect a big number on their customer scale. On the contrary, the number is not always uniform. Without analysis, the hotel tends to assign a lot of employees during the off-peak hours. Hence, the food prepared, and the human resource assigned during the minimal visit time zone gets wasted. Additional employees result in additional wage expenses to the management. The vice-versa of this situation is also possible. Number of waiters to attend to the customers during peak hours might be less. All these are the results of lack of analysis based on customer density.

2.3.2. Proposed Solution

To overcome this inequality between customers and resources, we propose a detailed survey to determine the number of customers in the restaurant during different periods of the day, week, month and year. Based on this survey, the management can plan ahead on the number of employees and their work shifts required during the day. This analysis helps the management in efficiently and economically allocating the resources on a day to day basis, and to also plan to meet future needs based on the trend observed.

2.3.3. Plan of work

Anvitha and Priyanka will implement the following strategy -

- Collect data on the number of customers visiting the restaurants in a day (morning/afternoon/evening/night), a week, a month and a year.
- Analyze the types of customers visiting the restaurants; adults, children, vegetarians, non-vegetarians.
- Using the gathered information, data analytics is performed to calculate the average number of employees required during the different working hours of the day, and the approximate kind of food that must be prepared to satisfy the needs of the visiting customers. This helps the management reduce human resources as well as monetary resources.

The management can also use the results of the analysis to improve different aspects of their business. For instance, if the restaurant is not attracting as many customers as they would during a particular period, they may conduct a survey to determine why customers do not frequent the restaurant during those periods and find a solution to bridge the gap between the customers' needs and the restaurants' service. In this way, customer relations as well as the quality of service can be enhanced, thereby improving the business which leads to increased profits.

2.4. Item Based Revenue Maximization

2.4.1. Problem Statement

Currently, restaurant managers do not have statistics as to how the various items in their menu sells. It is assumed that every item on the menu will be ordered almost the same number of times, except for one or two 'customer favorites'. There is no store specific data to assess the items that are being sold and their exact volumes. Managers are not wary of the demands that may arise for a particular item in the near future, hence the quality of service will take a hit when such a situation arises where a certain item is in high demand but the kitchen is not equipped to supply the same. On the contrary, there are a few items on the menu that do not sell at all or sell very less. The restaurant ends up wasting a substantial amount of resources in trying to keep that item 'ready to serve'.

2.4.2. Proposed Solution

We propose to aid the managers in determining trends in the way different items in the restaurant menu sell, so that they are prepared for situations like a sudden hike in demand for a particular item. This way they will not lose out on revenue that could have been generated if the kitchen could keep up with the demand. On hindsight, the restaurant can also avoid wasting resources by preparing items that are seldom sold.

2.4.3. Plan of Work

Ashwin and Suraj will use the following approach:

- Talk to restaurant managers and enquire about the obstacles they currently encounter while preparing and selling various items in the menu.
- Collect data on the various items on the menu and the number of portions sold throughout the week.
- Determine the time of day(morning/noon/evening) during which each item is sold the most and least.
- In the coming weeks, use data analysis to determine trends and patterns in the way various items in the menu sell using the data from the restaurant's sales database.
- With this organized information, the future trends in restaurant sales can be determined beforehand which will be a great asset for restaurant operation.

3. GLOSSARY OF TERMS

3.1. Technical Terms

Database: The structured set of data collected and stored based on food items, inventory, customer profile specific to different locations.

Graphical user interface: The interface that allows the user to interact with the application through graphical icons and visual indicators.

Data Sets: Collection of data that corresponds to the contents of a single database table.

Data Analytics: The process of analyzing the collection of data to draw useful conclusions and make informed business decisions.

3.2. Non- Technical Terms

Inventory: The list of food items and the ingredients used to prepare the food.

Geographical location: The state wise location of different branches of the same restaurant.

Popularity index: To indicate the frequency of food items ordered.

Customer Density: The number of customers visiting the restaurant during a interval of time.

Off peak hours: The period when the restaurant experiences a lower customer density.

Human resources: The number of workers employed.

Ready to serve: The prepared food that is available for immediate consumption by the customer.

4. SYSTEM REQUIREMENTS

Table 1- System Requirements

Requirements	Priority	Description
RQ 1	5	The system shall obtain data from the POS system throughout the day and store it in the database.
RQ 2	4	The system shall retrieve item wise sales data from the database to analyze the trend in sales of various items.
RQ3	3	The system will prioritize the items based on the frequency of sales throughout the day for the entire week.
RQ4	4	The system will predict and suggest the items which may be in high/low demand based on time of day
RQ5	4	The system will retrieve data on the number and type of customers visiting the restaurant during different period of the day, month, year.
RQ6	3	The system will analyze and determine the peak/off-peak working hours, demand on the type of food.
RQ7	4	The system will provide the minimum number of workers required during a shift, suggest new additions in menu, changes in ambience and introducing new offers.
RQ8	4	The system will retrieve the data based on location of the restaurant and its popularity index.
RQ9	3	The system will prioritize the food items based on its popularity at specific locations.
RQ10	4	The systems will recommend the changes in menu, price reductions and suggest special offers.
RQ11	4	The system shall use the inventory to suggest quantity of food to be added to the quotation for the month.
RQ12	3	The system shall keep track of the older products on the shelf and push them forward for usage before replacing with newer ones.
RQ13	4	The system shall consider the rate of usage for each ingredient before making a quotation.

5. DATA SETS

We will create a customized database based on the combination of the existing datasets of restaurants available online.

We will primarily have 2 datasets -

1) User Dataset – This set will have Food items, Price, Discount, Type of Customer (Based on Age), Location of the restaurant, Time of Order as the columns.
2) Inventory Dataset – This set will have details of individual raw materials, Quantity left, Total Quantity, Perishable items.

Part 1 – Creating the data sets

We will use XAMPP server which has support for both PHP and MySQL. We will create a portal in PHP from where users can select the food items. For each selection we will embed this choice with the location information, time of the day, Price, discount and it will be stored to the database as a single entry.

Part 2 – Accessing the datasets and analyzing information

We will use python for data analytics. We will access the database from the "sql-connector" package available in python.

6. USER STORIES

As a Restaurant Manager:

Table 2- User Stories: Restaurant manager

Identifier	User Stories	Size
ST1	I can determine who has access to the point of sales system.	3
ST2	I can determine who has access to the trends obtained from restaurant data analytics.	3
ST3	I can modify the menu based on the current trends recorded.	6
ST4	I can modify the price of various items on the menu and work on restaurant offers.	6
ST5	I can decide the number of employees on duty for a particular time of the day and day of the week.	5
ST6	I have access to the sales, inventory and employee database of the restaurant for verification and validation.	4
ST7	I decide on the inventory to be ordered based on requirements from the kitchen.	5
ST8	I have access to the profit/loss statements of the restaurant.	5

As a chef:

Table 3- User Stories: Chef

Identifier	**User Stories**	**Size**
ST1	I have access to the item inventory database.	3
ST2	I can request/approach the restaurant manager regarding changes in the menu items based on usage.	5
ST3	I can reason with the manager to make sure of the feasibility of a menu change decision.	4
ST4	I have access to the anticipated trends in item sales.	3
ST5	I can optimize inventory usage to minimize waste.	6

As a Waiter:

Table 4- User Stories: Waiter

Identifier	**User Stories**	**Size**
ST1	I am responsible for entering the orders into the point of sales system	5
ST2	I am responsible for tallying the current order and the items served on the table	4

7. FUNCTIONAL REQUIREMENT SPECIFICATION

7.1 Stakeholders

Stakeholders are the people interested in the success of the organization. They are classified as primary and secondary stakeholders. In our case, the primary stakeholders are the managers, host and the other staffs working in the restaurant who will be directly utilizing the system. The Another set of stakeholders will be the customers who don't directly use the system by themselves, but their feedback and experience plays an important role in the improvement of the organization.

7.2. Actors and Goals

Manager

Role: The person who manages the entire restaurant.

Goals: Manages inventory, payroll, employee scheduling.

Host

Role: The person who interacts with the customer and waiter.

Goals: Seats guests and assigns waiter to tables

Waiter

Role: The employee who interacts with and serve dine-in customers.

Goals: Takes customers' orders to the kitchen and delivers the prepared order to the customer

Cook

Role: The person who oversees the kitchen and preparing meals.

Goals: Reads the order details received from the supervisor, prepares the food, and informs the supervisor when it is ready

Busboy

Role: The person who is responsible for the cleanliness of the restaurant.

Goals: Keeps track of which tables are being used; cleans tables and updates their status as necessary

7.3. Use Cases

7.3.1 Casual Description

1) **UC1 – To place the order**. This use case will involve displaying the menu portal using the menu table from the database. It will also detect the user selection food item and add a new entry into the Users database.
2) **UC2 – Dataset Creation.** This use case will involve the method to create the artificial dataset based on the mathematical model.
3) **UC3 – Clustering.** This use case will involve the method to cluster various parameters from the dataset according to the different analysis modules. The data set will contain the set of all the parameters and only the required parameters must be clustered according to the need.
4) **UC4 – Analyze trend in sales based on the time**. This will involve keeping a track of the frequency of sales at various times during the day and predict increase/decrease in demand at specific hours.
5) **UC5 – Analysis based on Customer density**. This will involve keeping track of the type of customers based on age, interests and correlating it with the food items purchased.

Finally, it will make a prediction about the type of food items preferred by specific type of customers and notify about menu updates required if any to the management.

6) **UC6 – Trend analysis based on Geography**. This use case involves keeping a track of the popularity of food items across different locations of the restaurant. It will then make predictions about the change in menu, price modifications or offers based on the results of classification.
7) **UC7 – Inventory management**. This will involve keeping a track of the raw materials in the inventory based on the customer selection of food items. It will then predict the required updates of raw materials in the inventory.
8) **UC8 – Maintaining Statistics**. This involves combining the results of all the data analytics features and providing statistical reports about this data to the restaurant manager.

7.3.2 Use Case Diagram

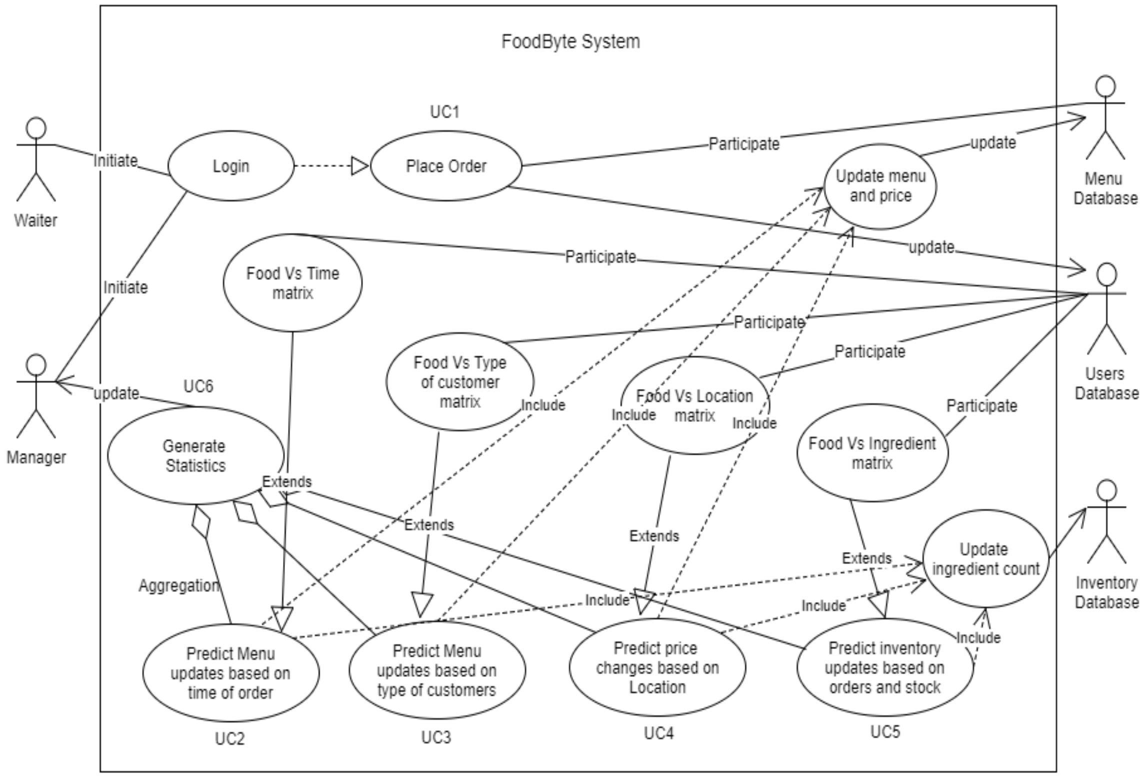


Figure 1: Use Case diagram

7.3.3 Traceability Matrix

Table 5- Traceability Matrix

	UC1	UC2	UC3	UC4	UC5	UC6	UC7	UC8
RQ1	**X**	**X**						
RQ2	**X**		**X**	**X**				
RQ3				**X**				
RQ4				**X**				**X**
RQ5	**X**			**X**	**X**			
RQ6				**X**	**X**			
RQ7					**X**			**X**
RQ8	**X**					**X**		
RQ9						**X**		
RQ10						**X**		**X**
RQ11	**X**						**X**	
RQ12							**X**	
RQ13							**X**	**X**

7.3.4. Fully Dressed Use Cases

Use Case 1: To place an Order

Table 6- UC1

USE CASE UC1	**To Place an Order**
Related Requirements	RQ1, RQ2, RQ5, RQ8, RQ11 from Table1
Initiating Actor	Waiter
Actor's Goal	To take the order from the customer and select it in the Food Byte portal
Participating Actors	Menu Database, Users Database
Preconditions	- Menu Database should contain the updated values - The Food Byte portal should be synced with the Menu Database

Postconditions	The User Database should be updated with the Customer selection and notification to be sent to the Data analytics modules to start processing
Flowof Events for Main Success Scenario	1. Waiter enters login information into the portal 2. System retrieves the Menu from the Menu Database and displays it on the portal 3. Waiter enters the customer selection and an entry is made in the User Database 4. System signals the Data analytics module about the new entry.

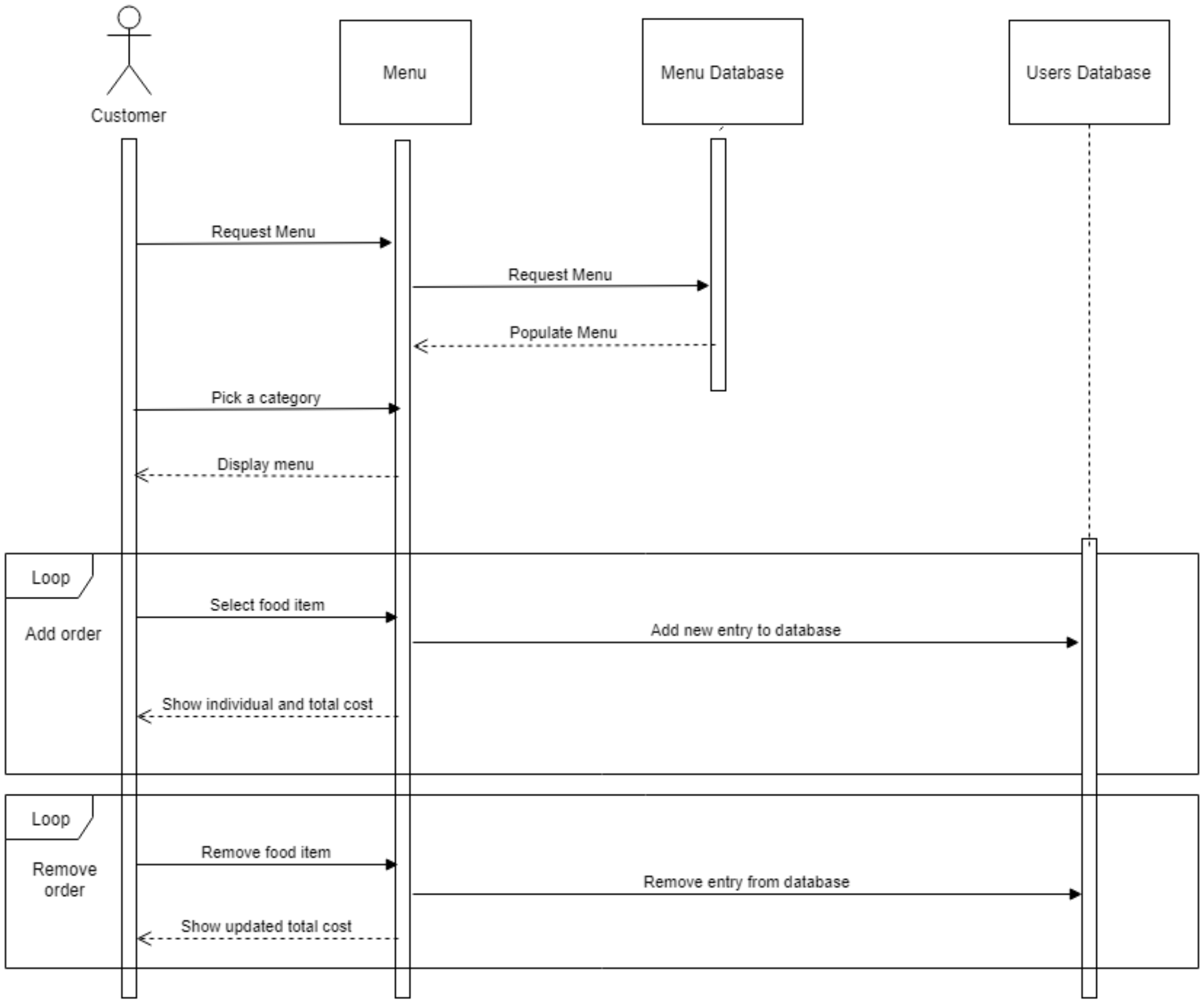


Figure 2: To place an order

Use case 2: To create an Artificial Dataset

Table 7- UC2

USE CASE UC2	**To create an artificial dataset**
Related Requirements	RQ 1 from Table1
Initiating Actor	Food Byte System
Actor's Goal	To provide the dataset with all the parameters required for efficient working of the system
Participating Actors	Users Database
Preconditions	- The mathematical model to create the artificial dataset should be predetermined.
Postconditions	The User Database should be updated with theCustomer selection and notification to be sent to theData analytics modules to start processing
Flow of Events for Main Success Scenario	1. The reliable value is read from the csv file 2. According to the mathematical model used the values are estimated and dataset tables are created 3. The different tables created for different parameters are merged together to form the database

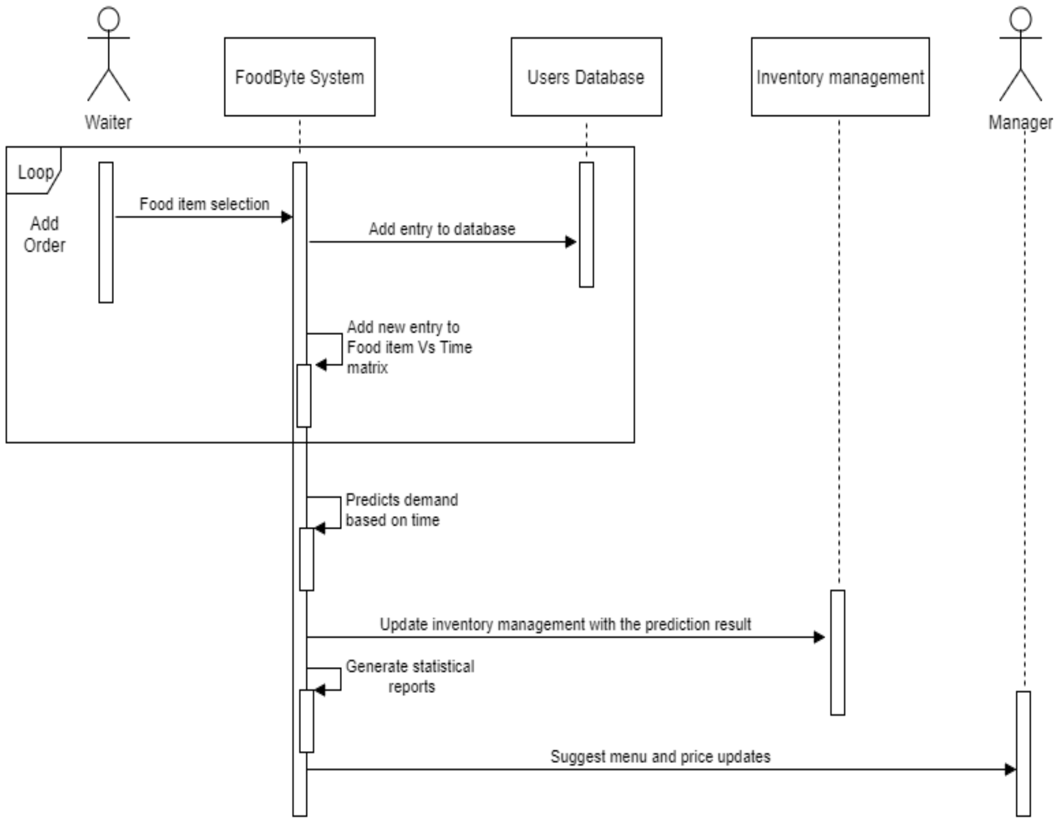

Waiter
FoodByte System
Users Database
Inventory management
Manager
Loop
Add Order
Food item selection
Add entry to database
Add new entry to Food item Vs Time matrix
Predicts demand based on time
Update inventory management with the prediction result
Generate statistical reports
Suggest menu and price updates

Figure 3: To create an artificial dataset

Use case 3: Clustering of Data

Table 8- UC3

USE CASE UC3	**Clustering of Data**
Related Requirements	RQ2, RQ3, RQ5,RQ8, from Table1
Initiating Actor	FoodByte system
Actor's Goal	To retrieve entries from the Users Database and cluster them into groups based on location and sub groups based on customer density, item sales and ethnicity.
Participating Actors	Users Database
Preconditions	- Users Database should be updated with latest entries.
Postconditions	The system should provide Food item vs Time, customer density vs Time and Ethnicity data for further

	analysis.
Flow of Events for Main Success Scenario	1. The system receives notification about new entries from portal 2. System retrieves the required dataset from the Users database 3. The system clusters the data based on zone, ethnicity, customer density vs time and food item vs time 4. The System updates the data into the Clustered Database.

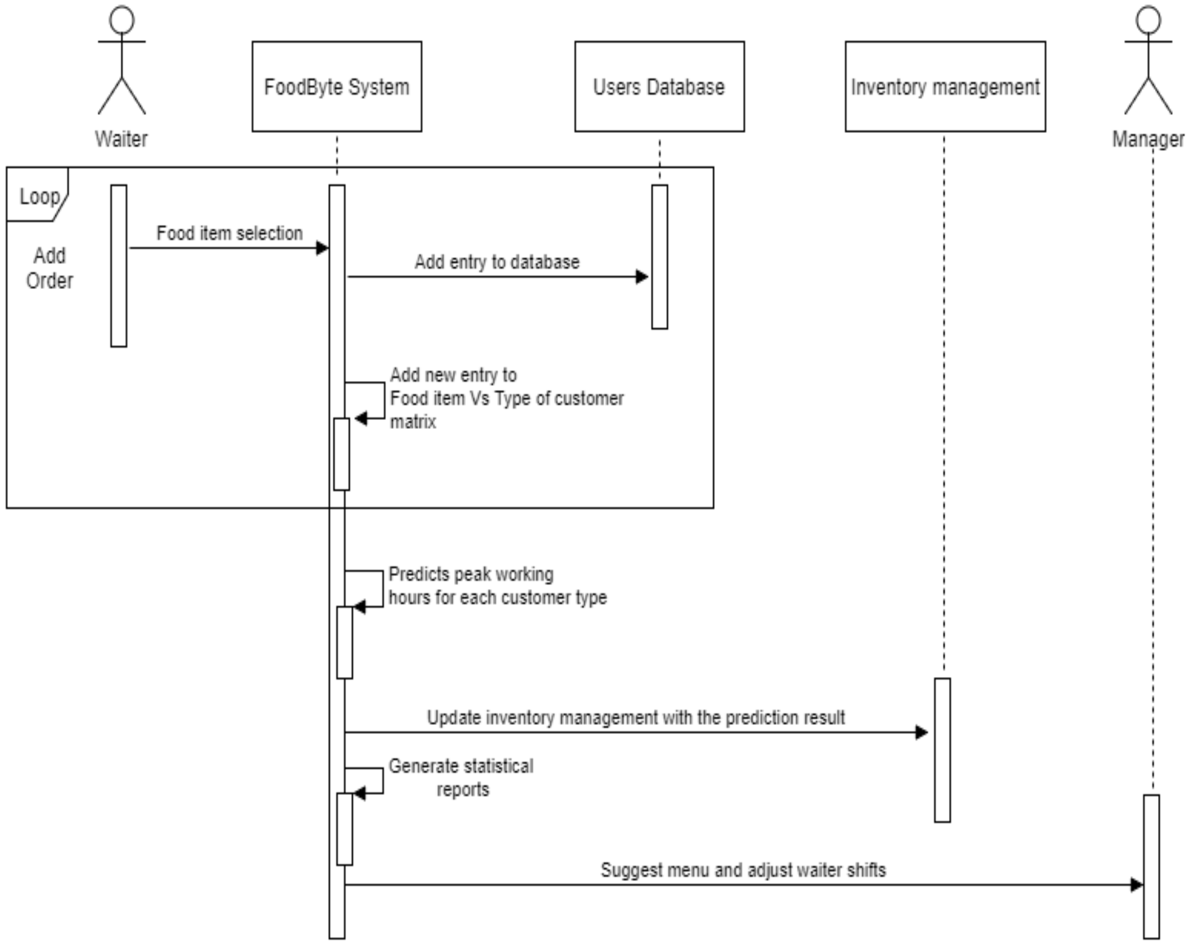
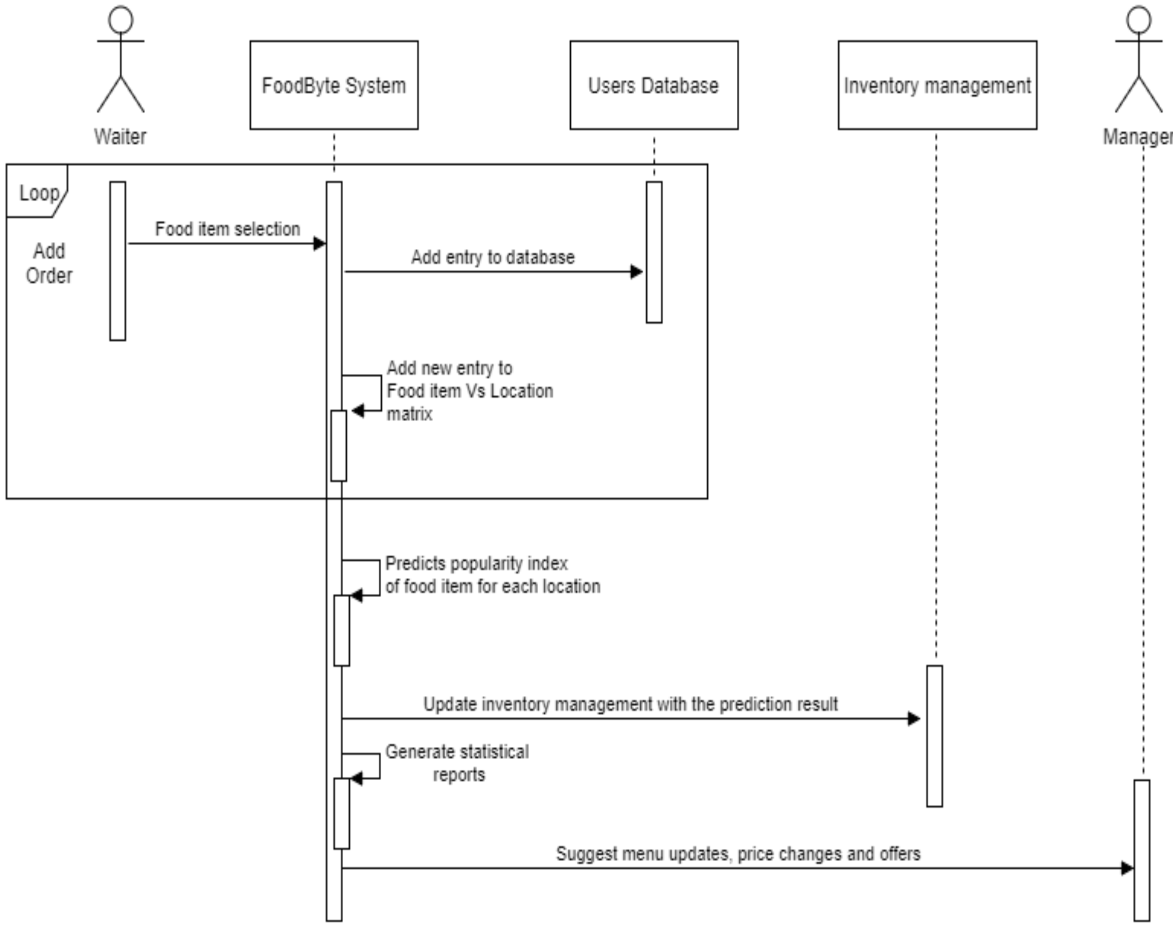

Waiter
FoodByte System
Users Database
Inventory management
Manager
Loop
Add Order
Food item selection
Add entry to database
Add new entry to Food item Vs Type of customer matrix
Predicts peak working hours for each customer type
Update inventory management with the prediction result
Generate statistical reports
Suggest menu and adjust waiter shifts

Figure 4: Clustering of Data

Use Case 4: Trend Analysis based on Geography

Table 9- UC4

USE CASE UC4	**Trend Analysis in sales based on time**
Related Requirements	RQ2, RQ3, RQ4, RQ5, RQ6 from Table1
Initiating Actor	FoodByte system
Actor's Goal	To retrieve entries from the Users Database andprovide sales predictions at specific times
Participating Actors	Users Database, Inventory Database
Preconditions	- Users Database should updated with latest entries - The system should have the Food item vs time dataset in the required form
Postconditions	The system should use the Food item vs Time matrixfrom the dataset and provide predictions about thedemand at specific times during the day
Flow of Events for Main Success Scenario	5. The system receives notification about new entries from portal 6. System retrieves the required dataset from the Users database 7. System predicts changes in demand during the day, informs inventory and suggests menu updates based on these predictions. 8. System will consolidate these predictions to display statistical reports



Waiter
FoodByte System
Users Database
Inventory management
Manager
Loop
Add Order
Food item selection
Add entry to database
Add new entry to Food item Vs Location matrix
Predicts popularity index of food item for each location
Update inventory management with the prediction result
Generate statistical reports
Suggest menu updates, price changes and offers

Figure 5: Sales analysis based on time

Use Case 5: Analysis based on Customer Density

Table 10- UC5

USE CASE UC5	**Analysis based on Customer density**
Related Requirements	RQ5, RQ6, RQ7 from Table1
Initiating Actor	FoodByte system
Actor's Goal	To retrieve entries from the Users Database and suggest menu updates based on type of customers
Participating Actors	Users Database
Preconditions	- Users Database should updated with latest entries - The system should have the Food item vs Type of Customer dataset

Postconditions	The system should use the Food item vs Type ofcustomer matrix to predict peak working hours andsuggest menu updates at those times
Flowof Events for Main Success Scenario	1. The system receives notification about new entries from portal 2. System retrieves the required dataset from the Users database 3. System predicts the peak working hours during the day based on the type of customers and suggest menu updates based on these predictions 4. System will consolidate these predictions to display statistical reports

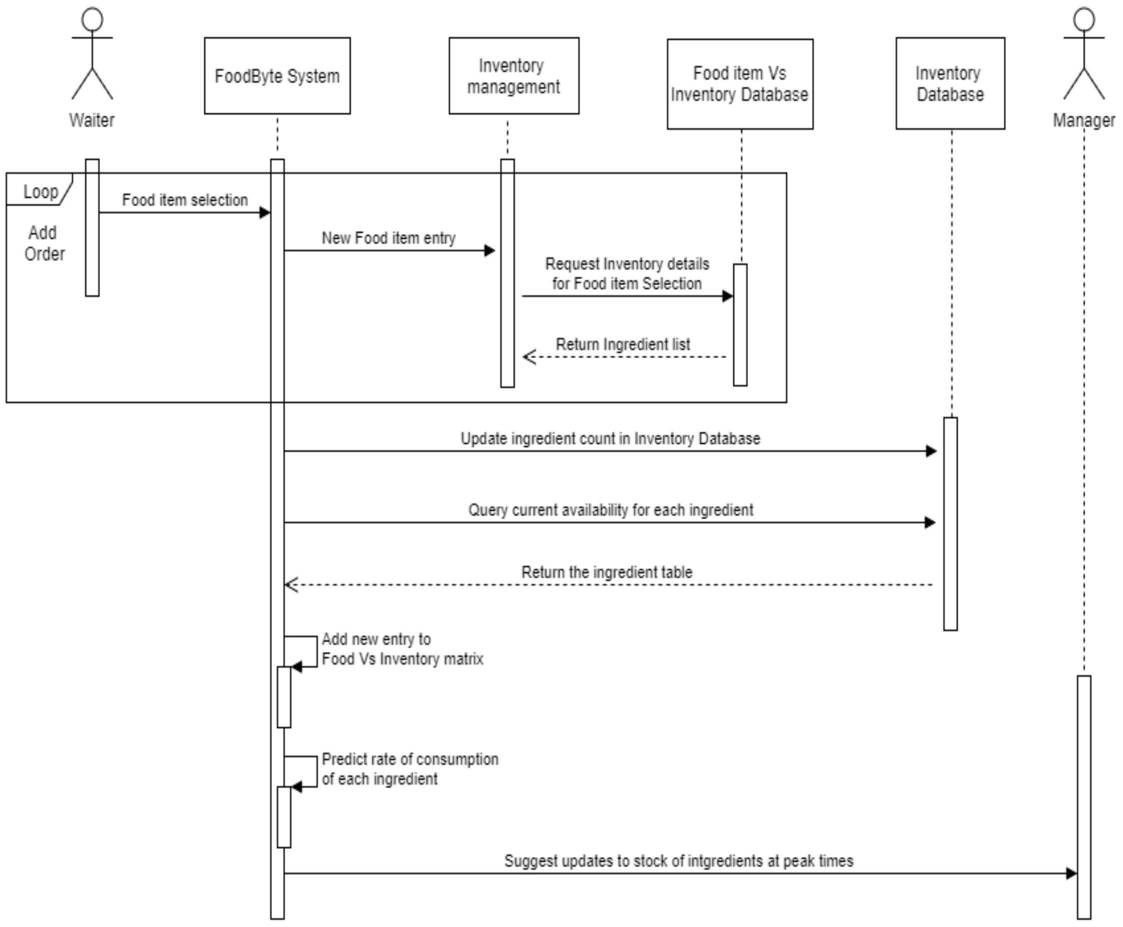

Waiter
FoodByte System
Inventory management
Food item Vs Inventory Database
Inventory Database
Manager
Loop
Add Order
Food item selection
New Food item entry
Request Inventory details for Food item Selection
Return Ingredient list
Update ingredient count in Inventory Database
Query current availability for each ingredient
Return the ingredient table
Add new entry to Food Vs Inventory matrix
Predict rate of consumption of each ingredient
Suggest updates to stock of intgredients at peak times

Figure 6: Analysis based on customer density

Use Case 6: Trend Analysis based on Geography

Table 11- UC6

USE CASE UC6	**Trend Analysis based on Geography**
Related Requirements	RQ8, RQ9, RQ10 from Table1
Initiating Actor	Food Byte system
Actor's Goal	To retrieve entries from the Users Database and suggest price changes based on location
Participating Actors	Users Database, Inventory Database
Preconditions	- Users Database should updated with latest entries - The system should have the Food item vs Type of Location dataset
Postconditions	The system should use the Food item vs Location matrix and predict the price changes or special offers needed at specific locations of the restaurant
Flowof Events for Main Success Scenario	1. The system receives notification about new entries from portal 2. System retrieves the required dataset from the Users database 3. System predicts the popularity index of food items at specific locations of the restaurant and use it to suggest price changes or special offers on specific food items 4. System will notify the inventory management system about the changes in demand of food items at that location 5. System will consolidate these predictions to display statistical reports

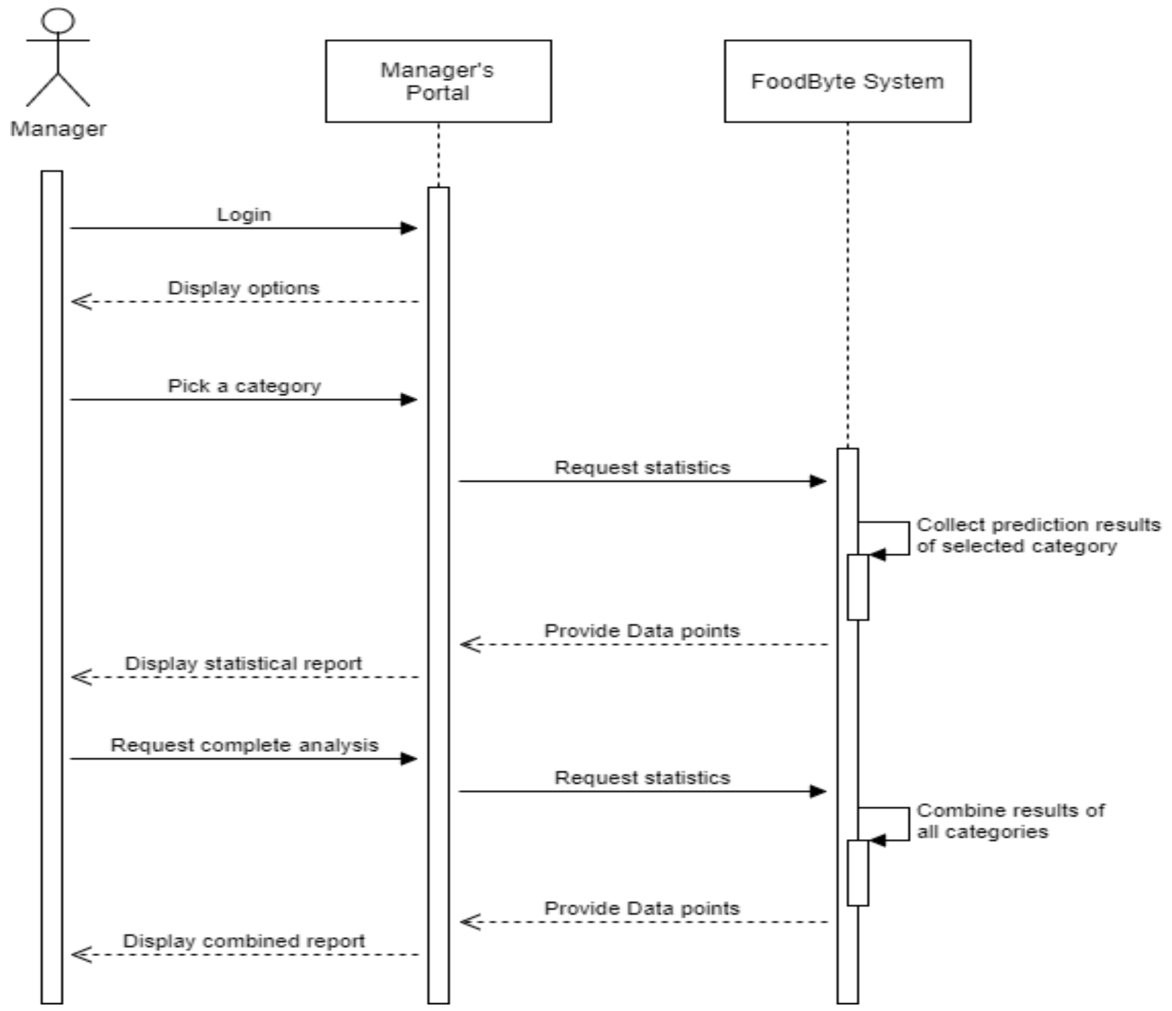


Figure 7: Trend analysis based on Geography

Use Case 7: Inventory Management

Table 12- UC7

USE CASE UC7	**Inventory Management**
Related Requirements	RQ11, RQ12, RQ13 from Table1
Initiating Actor	Food Byte system
Actor's Goal	To retrieve entries from the Inventory Database and predict the required raw materials based on demand
Participating Actors	Users Database, Inventory Database
Preconditions	- Inventory Database should updated with latest entries - The system should have the Food item vs Inventory dataset

Postconditions	The system should use the Food item vs Inventory matrix, predict the rate of consumption of an ingredient and allow the restaurant to maintain the stock at peak times
Flow of Events for Main Success Scenario	1. The system receives notification about new entries from portal 2. System retrieves the required dataset from the Inventory database 3. System predicts the rate of consumption of an ingredient and notify the restaurant management about the availability of popular ingredients. 4. System will consolidate these predictions to display statistical reports

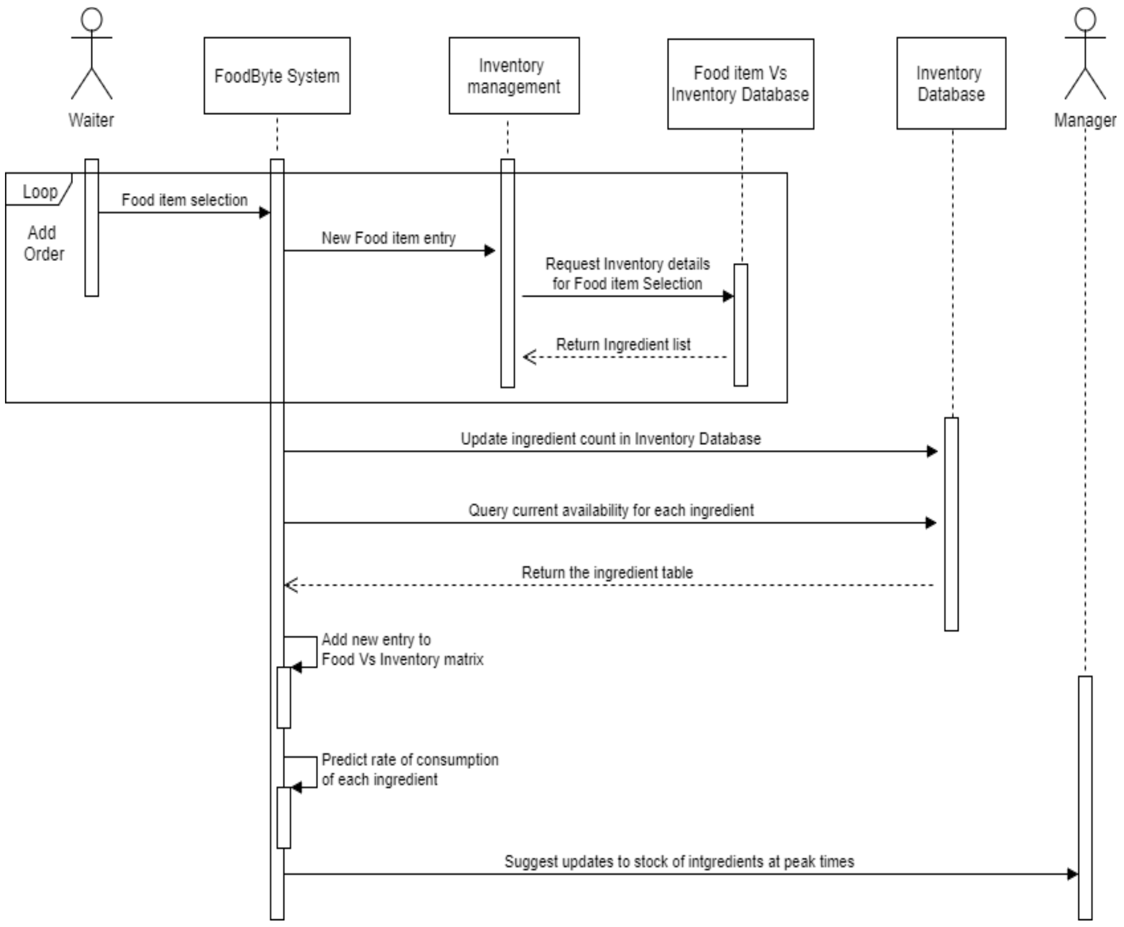

Waiter
FoodByte System
Inventory management
Food item Vs Inventory Database
Inventory Database
Manager
Loop
Add Order
Food item selection
New Food item entry
Request Inventory details for Food item Selection
Return Ingredient list
Update ingredient count in Inventory Database
Query current availability for each ingredient
Return the ingredient table
Add new entry to Food Vs Inventory matrix
Predict rate of consumption of each ingredient
Suggest updates to stock of intgredients at peak times

Figure 8: Inventory management

Use Case 8: Maintaining Statistics

Table 13- UC8

USE CASE UC8	**Maintaining Statistics**
Related Requirements	RQ4, RQ7, RQ10, RQ13 from Table1
Initiating Actor	FoodByte system
Actor's Goal	To create the statistical report based on the predictions of UC2 – UC5
Participating Actors	Manager
Preconditions	- UC2 – UC5 should have completed their predictions - The system should be able to generate a UI based report
Postconditions	The system should integrate the individual predictions of UC2 – UC5 into a single conclusion
Flow of Events for Main Success Scenario	1. The Manager logs into the FoodByte portal and requests a business report 2. System integrates the predictions of all the modules into a conclusion. 3. System displays a business report to the manager and provides multiple suggestions based on the statistics

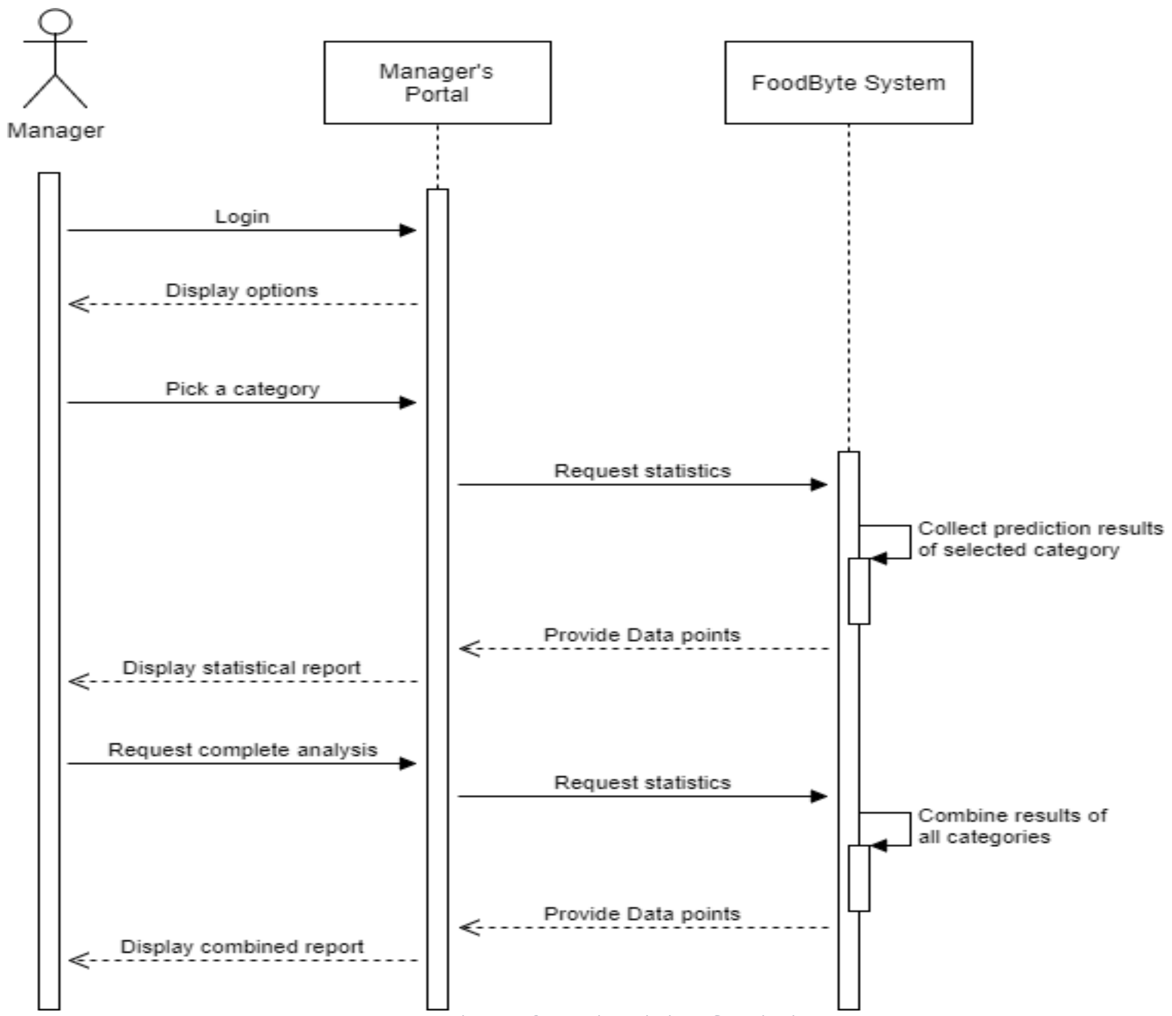


Figure 9: Maintaining Statistics

8. EFFORT ESTIMATION USING USE CASE POINTS

Actor Classification:

Table 14

Actor Name	Description of relevant characteristics	Complexity	Weight
Database	Database is a system interacting through a protocol	Average	2
Manager	Manager interacts with the system through a graphical user interface to view analysis results and take administrative actions	Complex	3

Waiter	Waiter interacts with the system through a graphical user interface to add order entries	Simple	1
Inventory	Inventory interacts with the system and the Database to make predictions and update	Average	2

Unadjusted Actor Weight is calculated by,

UAW = 1 × Simple + 2 × Average + 1 × Complex = (1 × 1) + (2 × 2) + (1 × 3)
UAW = 8

Use Case classification:

Use Case	**Description**	**Category**	**Weight**
Place Order (UC – 1)	Simple User Interface. 4 Steps for main success scenario. Database is the participating actor.	Simple	5
Dataset Creation (UC – 2)	FoodByte system must create table entries for each of the use cases UC-4 to UC-6, merge all the table entries based on the mathematical model, and update the database. Database is the participating actor.	Complex	15
Clustering (UC – 3)	FoodByte system performs nested clustering on each entry based on food category, location, ethnicity, customer density and time to classify the data into these clusters and sub-clusters required for analysis and prediction. Database is the participating actor.	Complex	15
Trend Analysis in sales based on Time (UC – 4)	FoodByte system retrieves Food category Vs Time clusters, sorts food items from maximum to minimum demand, informs inventory, and suggests updates based on results. Database is the participating actor.	Complex	15
Customer Density Analysis (UC – 5)	FoodByte system retrieves Time Vs Customer density clusters, sorts working hours in order of maximum to minimum number of customers, and provides suggestions based on results. Database is the participating actor.	Complex	15
Trend Analysis based on Geography (UC – 6)	FoodByte system retrieves Food item Vs Location clusters, sorts items in each category from maximum to minimum popularity at each location, suggests price changes or special offers on specific food items, and notifies inventory of changes in demand. Database is the participating actor.	Complex	15

Inventory Management (UC – 7)	FoodByte system retrieves required dataset, predicts rate of consumption of each ingredient and notifies management of availability of ingredients. Database is the participating actor.	Complex	15
Maintaining Statistics (UC – 8)	FoodByte system integrates predictions of UC-4 to UC-7 into a conclusion and displays a business report to Manager and provides multiples suggestions. Manager is the participating actor.	Average	10

Table 15

Unadjusted Use Case Weight is calculated by,

UUCW = (1 × Simple) + (1 × Average) + (6 × Complex)

= (1 × 5) + (1 × 10) + (6 × 15)

UUCW = 105

Technical Complexity Factors:

Table 16

Use Case	**Description**	**Weight**	**Perceived Complexity**	**Calculated Factor (Weight × Perceived Complexity)**
T1	Distributed System: Distributed Web-based system.	2	3	6
T2	Response time/Performance Objectives: Users expect moderately good performance. Nothing exceptional	1	3	3
T3	End-user efficiency: Users expect exceptional End-user efficiency to view periodically updated analysis results.	1	5	5
T4	Internal processing complexity: Internal processing involves dataset creation, clustering and prediction, and is very complex.	1	5	5
T5	Code reusability: The code must be able to be used for a different restaurant chain with minimum modifications.	1	4	4
T6	Easy to install:	0.5	3	1.5

	The system is easy to install and is moderately important.			
T7	Easy to use: Ease of use is very important for the Manager to view analysis results and take actions based on results.	0.5	5	2.5
T8	Portability to other platforms: No portability concerns except to keep database vendor options open.	2	2	4
T9	System maintenance: Maintenance would be required to update features of FoodByte System.	1	3	3
T10	Concurrent/parallel processing: Concurrent use is not required.	1	0	0
T11	Security features: Security is not a concern.	1	0	0
T12	Access for third parties: No direct access for third parties.	1	0	0
T13	End user training: No unique training needs.	1	0	0
			Technical Factor Total:	34

Technical Complexity Factor is calculated by,

TCF = Constant-1 + Constant-2 × Technical Factor Total = $C_1 + C_2 \cdot \sum_{i=1}^{13} W_i \cdot F_i$

Constant-1 (C1) = 0.6

Constant-2 (C2) = 0.01

W_i = Weight of i^{th} technical factor

F_i = perceived complexity of i^{th} technical factor

TCF = 0.6 + 0.01 × 34

TCF = 0.94

Use Case Points is calculated by,

UCP = (UUCW + UAW) × TCF

= 113 × 0.94

UCP = 106.22

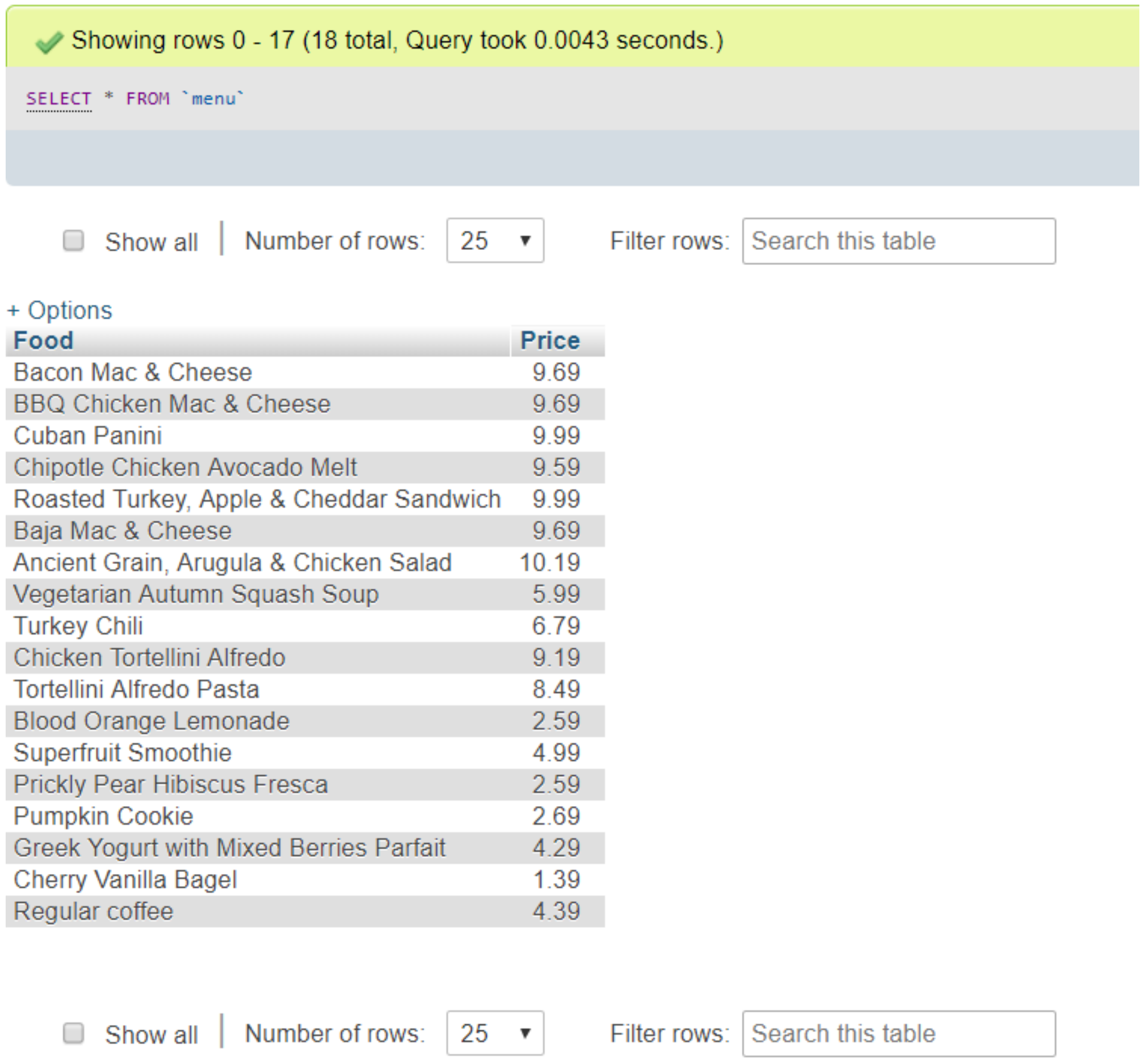
9.USER INTERFACE SPECIFICATION

9.1. Preliminary Design:

In this section the basic design of our project modules have been displayed as screenshots from our workplace.

Image 1: Displaying the menu

The food menu offered, and their prices will be displayed. Adding a menu or deleting a menu can be enabled by the manager.



```
✓ Showing rows 0 - 17 (18 total, Query took 0.0043 seconds.)

SELECT * FROM `menu`
```

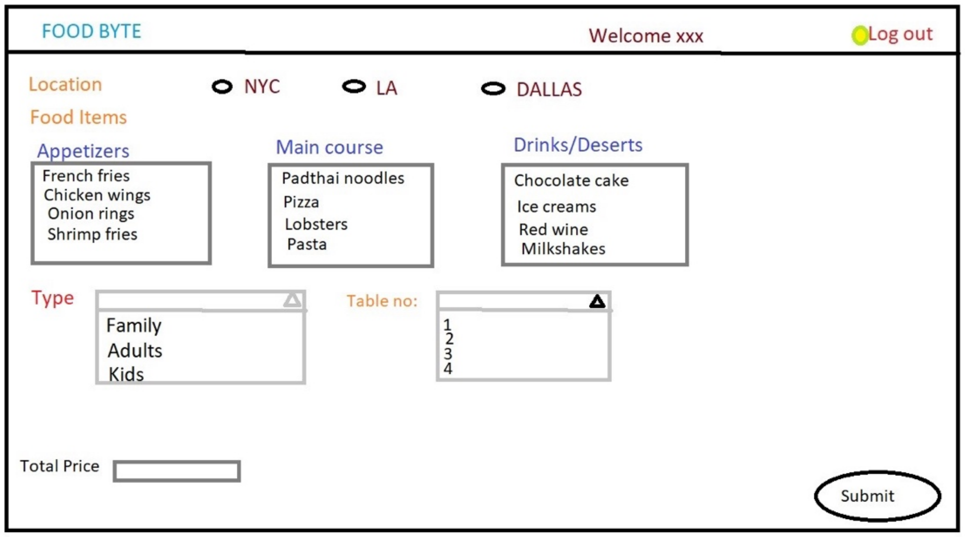
Food	Price
Bacon Mac & Cheese	9.69
BBQ Chicken Mac & Cheese	9.69
Cuban Panini	9.99
Chipotle Chicken Avocado Melt	9.59
Roasted Turkey, Apple & Cheddar Sandwich	9.99
Baja Mac & Cheese	9.69
Ancient Grain, Arugula & Chicken Salad	10.19
Vegetarian Autumn Squash Soup	5.99
Turkey Chili	6.79
Chicken Tortellini Alfredo	9.19
Tortellini Alfredo Pasta	8.49
Blood Orange Lemonade	2.59
Superfruit Smoothie	4.99
Prickly Pear Hibiscus Fresca	2.59
Pumpkin Cookie	2.69
Greek Yogurt with Mixed Berries Parfait	4.29
Cherry Vanilla Bagel	1.39
Regular coffee	4.39

Figure 10: Database menu display

The waiter will have a page to input the food ordered by the customer from the menu displayed. Multiple selections can also be done. The manager will have a page to check the statistical reports generate from each of the data analytics algorithms.

Waiter page

Figure 11: Menu selection page



Manager page

Figure 12: Manager page

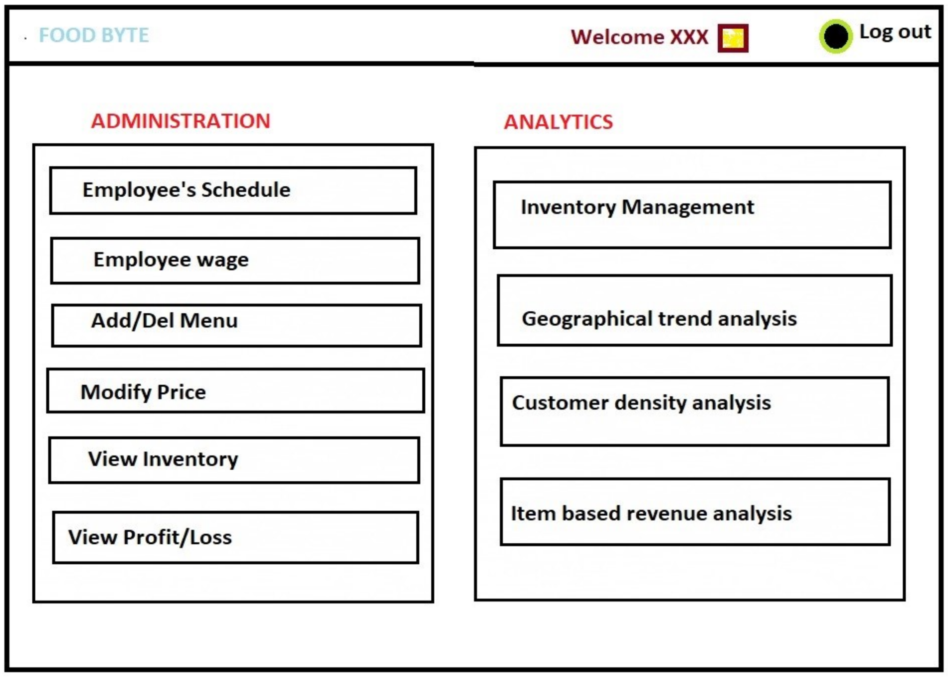
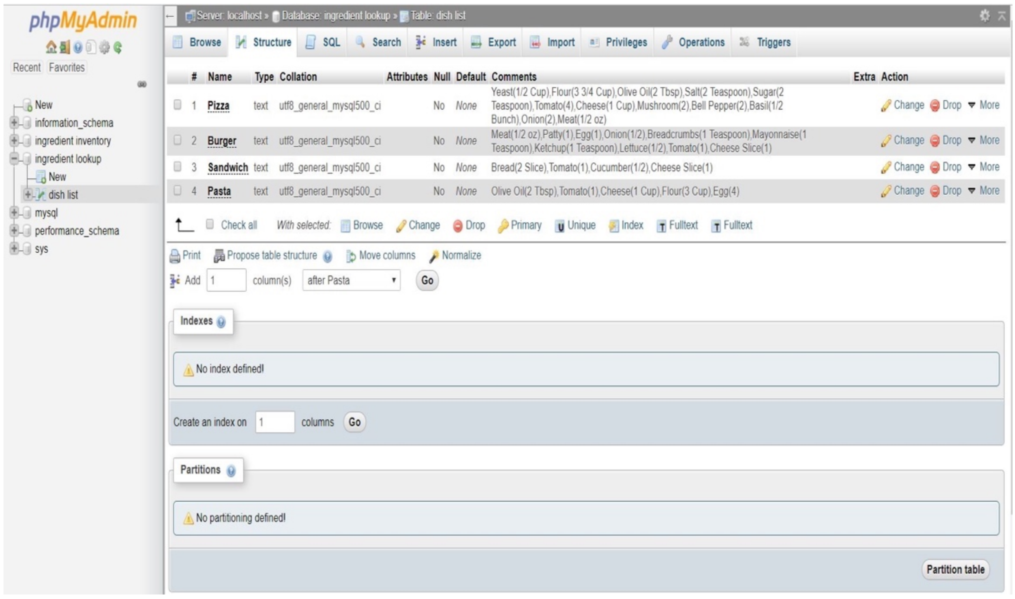


Image 3: Ingredient Lookup

The total number of food and the ingredients used for each food item is stored on a separate database.



Server: localhost » Database: ingredient lookup » Table: dish list

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	Pizza	text	utf8_general_mysql500_ci		No	None	Yeast(1/2 Cup),Flour(3 3/4 Cup),Olive Oil(2 Tbsp),Salt(2 Teaspoon),Sugar(2 Teaspoon),Tomato(4),Cheese(1 Cup),Mushroom(2),Bell Pepper(2),Basil(1/2 Bunch),Onion(2),Meat(1/2 oz)		Change Drop More
2	Burger	text	utf8_general_mysql500_ci		No	None	Meat(1/2 oz),Patty(1),Egg(1),Onion(1/2),Breadcrumbs(1 Teaspoon),Mayonnaise(1 Teaspoon),Ketchup(1 Teaspoon),Lettuce(1/2),Tomato(1),Cheese Slice(1)		Change Drop More
3	Sandwich	text	utf8_general_mysql500_ci		No	None	Bread(2 Slice),Tomato(1),Cucumber(1/2),Cheese Slice(1)		Change Drop More
4	Pasta	text	utf8_general_mysql500_ci		No	None	Olive Oil(2 Tbsp),Tomato(1),Cheese(1 Cup),Flour(3 Cup),Egg(4)		Change Drop More

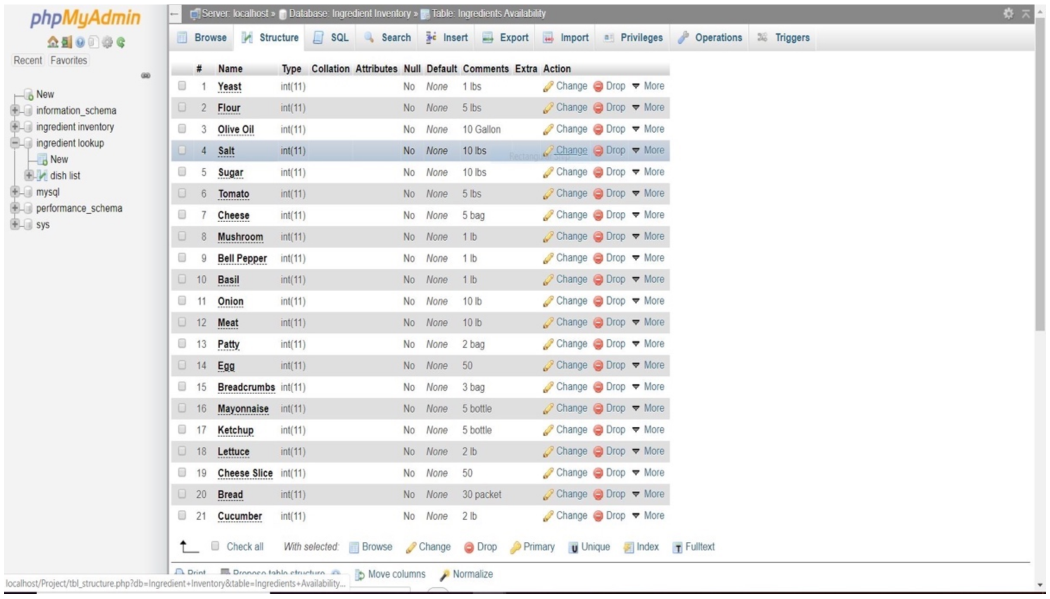
No index defined!

No partitioning defined!

Figure 13: Ingredient lookup

Image 4: Ingredient availability

After each item is ordered and prepared the used ingredients will be decremented from the inventory availability database. All this information will be needed for the data analytics.



Server: localhost » Database: Ingredient Inventory » Table: Ingredients Availability

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	Yeast	int(11)			No	None	1 lbs		Change Drop More
2	Flour	int(11)			No	None	5 lbs		Change Drop More
3	Olive Oil	int(11)			No	None	10 Gallon		Change Drop More
4	Salt	int(11)			No	None	10 lbs		Change Drop More
5	Sugar	int(11)			No	None	10 lbs		Change Drop More
6	Tomato	int(11)			No	None	5 lbs		Change Drop More
7	Cheese	int(11)			No	None	5 bag		Change Drop More
8	Mushroom	int(11)			No	None	1 lb		Change Drop More
9	Bell Pepper	int(11)			No	None	1 lb		Change Drop More
10	Basil	int(11)			No	None	1 lb		Change Drop More
11	Onion	int(11)			No	None	10 lb		Change Drop More
12	Meat	int(11)			No	None	10 lb		Change Drop More
13	Patty	int(11)			No	None	2 bag		Change Drop More
14	Egg	int(11)			No	None	50		Change Drop More
15	Breadcrumbs	int(11)			No	None	3 bag		Change Drop More
16	Mayonnaise	int(11)			No	None	5 bottle		Change Drop More
17	Ketchup	int(11)			No	None	5 bottle		Change Drop More
18	Lettuce	int(11)			No	None	2 lb		Change Drop More
19	Cheese Slice	int(11)			No	None	50		Change Drop More
20	Bread	int(11)			No	None	30 packet		Change Drop More
21	Cucumber	int(11)			No	None	2 lb		Change Drop More

localhost/Project/tbl_structure.php?db=Ingredient+Inventory&table=Ingredients+Availability...

Figure 14: Ingredient inventory

9.2. User Effort Estimation

Scenario 1: Waiter Places Order

a) Select the menu option.

b) Select the desired item.

c) Select the desired item, tap more than once to increase quantity

d) Once all desired items are in the order, select “place order” to add the order to the queue.

Scenario 2: Manager decides to view the trend analysis based on geography.

a) Select the option “trend analysis based on geography”

b) Select the item to remove it or modify its price based on the report.

c) Select the “total sales” option to view the sales

Scenario 3: Manager wants to build a quotation for required items for the following month.

a) Select the option “Inventory Available”

b) Select the option “Build quote for 1 month”

Scenario 4: Manager wants to build new menu based on inventory usage trend.

a) Select the option “Inventory”

b) Select the option “Insights”

c) Get the last month trend analysis of inventory list

d) Select “Suggest Menu”

Scenario 5: Manager wants to keep a track of the available Inventory.

a) Select the option “Inventory”

b) Select the option “Inventory Available” to view available resources

Scenario 6: Manager wants to keep a track of the popular items on the menu.

a) Select the option “sales database”.

b) Select the option “popularity”.

c) Enter the time to get time specific popularity in sales.

<u>Scenario 7: Customer takes survey form</u>

a) Enter basic details such as name, sex, age, vegetarian/non-vegetarian food preference
b) Select ‘Add member’
c) Repeat the same entries for all members of family/group visiting the restaurant
d) Provide feedback/comments in the ‘Customer Feedback’ box.
e) Click on ‘Submit’

<u>Scenario 8: Manager wants to modify shifts and wages of employees</u>

a) Select the tab ‘Analysis based on Customer Density’
b) Select the option ‘Employee Schedule’ to make changes to employee shifts
c) Select the option ‘Employee Salary’ to modify employee wages

<u>Scenario 9: Manager wants to make business decisions based on customer density</u>

a) Select the tab ‘Analysis based on Customer Density’
b) Select the option ‘Customer Survey Analysis’ to see peak/off-peak hours, type of customers and their food preferences
c) Make executive decisions to improve or change business practices based on suggestions provided by the system in the option ‘Recommendations and Tips’

10. DOMAIN ANALYSIS

10.1. Domain Model

a) Concept Definitions:

Table 17- Concept definitions

Responsibility Description	Type	Concept
R-01: Displays profit/losses, analysis of food item based on location and suggests modification of the menu accordingly.	D	FoodPopularity
R-02: Displays analysis based on customer density and suggests number of employees to be assigned.	D	CustomerDensity
R-03: Knows all orders from all customers/tables	K	OrderQueue

R-04: Views the items left in inventory and manages the inventory accordingly.	D	Manger
R-05: Place food orders within the restaurant	K	Waiter
R-06: Place drink orders within the restaurant	K	Waiter
R-07: System knows count of ingredients	K	Ingredientcount
R-08: Can view the statistics of the items sold and remove the least sold item	D	Manager
R-09: System displays the statistics about the ingredients in the inventory	K	Ingredientcount
R-10: System knows schedule of all employees	K	Schedule
R-11: Can edit expenses of store	D	Manager
R-12: System sends orders to the restaurant.	D	Order
R-13: Modify menu	D	MenuModifier
R-14: System knows schedule of all employees	K	Schedule
R-15: Can modify the item's price	D	Manager
R-16: Can provide discounts/offers on a particular item	D	Manger
R-17: Can provide info about place	K	Location
R-18: Can store info and update it	K	DatabaseConnection

b) Association Definitions:

Table 18 – Association definitions

Concept Pair	**Association Description**	**Associate Name**
Manager ↔ MenuModifer	Allows manager to modify menu	Modifies
Manager ↔ IngredientCount	Manager requests updates on IngredientCount to manage menu items	Requests Updates

Customer ↔ Waiter	Customer passes order requests to waiter	Conveys Requests
Manager ↔ Schedule	Schedule passes employee’s schedule requests to manager	Conveys Requests
Manager ↔CustomerDensity	Allows manager to view customer density statistics	Provides Data
Manager ↔FoodPopularity	Allows manager to view popular/least selling items	Provides Data
Waiter ↔Order	Allows the waiter to take order from customer	Provides Data
Waiter ↔ Location	Allows waiter to select location of restaurant	Provides Data
Manager↔DatabaseConnection	Allows manager to view and update information correctly	Provides Data

c) Attribute Definitions:

Table 19- Attribute definitions

Concept	**Attribute**	**Attribute Description**
Manager	Name	Name of Manager
	Manager ID	Manager has a unique identification credential associated with login
	Privileges	Manager has specific privileges including menu modifications, modify number of waiters working at a particular time, view different statistics related to sales, inventory etc, profit/loss, etc.
Waiter	Name	Name of the waiter
	Waiter ID	Each waiter has a unique identification credential associated with them
FoodPopularity	Favourite	The most sold item of the restaurant
	Least popular	The least sold item

OrderQueue	Orders	Orders are placed on a queue within the system.
Order	Customer Information	Information about the customer who placed the order.
	Location	The current location of the restaurant
	Food item	The item ordered by the customer
Schedule	Date	The various dates available on the schedule
	Time	The time slots available on each day
	Employee Name	Name of the employee working a certain shift
IngredientCount	Name	Name of the ingredient
	Total Stock	Amount of stock available in the kitchen for each ingredient
MenuModifier	Name of Item	Allows for altering the name of the selected item
	Ingredients	Allows for altering ingredients of a selected item
	Price	Allows for altering the price of a selected item
CustomerDensity	Number of customers	Tracking the number of customers at particular time during the day
	Number of employees	Used to determine the number of employees required for a particular shift based on the customer density
	Time	The time slots available on each day
DatabaseConnection	Status	Shows if connected or not in order to update information correctly
Location	Name	Provides crucial information about the location for data analytics

10.2 Traceability Matrix

Table 20- Traceability Matrix

	Domain Concepts										
Use Case	FoodPopularity	CustomerDensity	OrderQueue	Manager	Waiter	IngredientCount	Schedule	Order	Location	DatabaseConnection	MenuModifier
UC1			X		X			X			
UC2		X						X	X	X	
UC3	X	X						X	X	X	
UC4	X			X	X		X				X
UC5		X		X	X		X				X
UC6				X					X		X
UC7	X	X		X		X			X		
UC8				X						X	

10.3 Domain Model Diagram

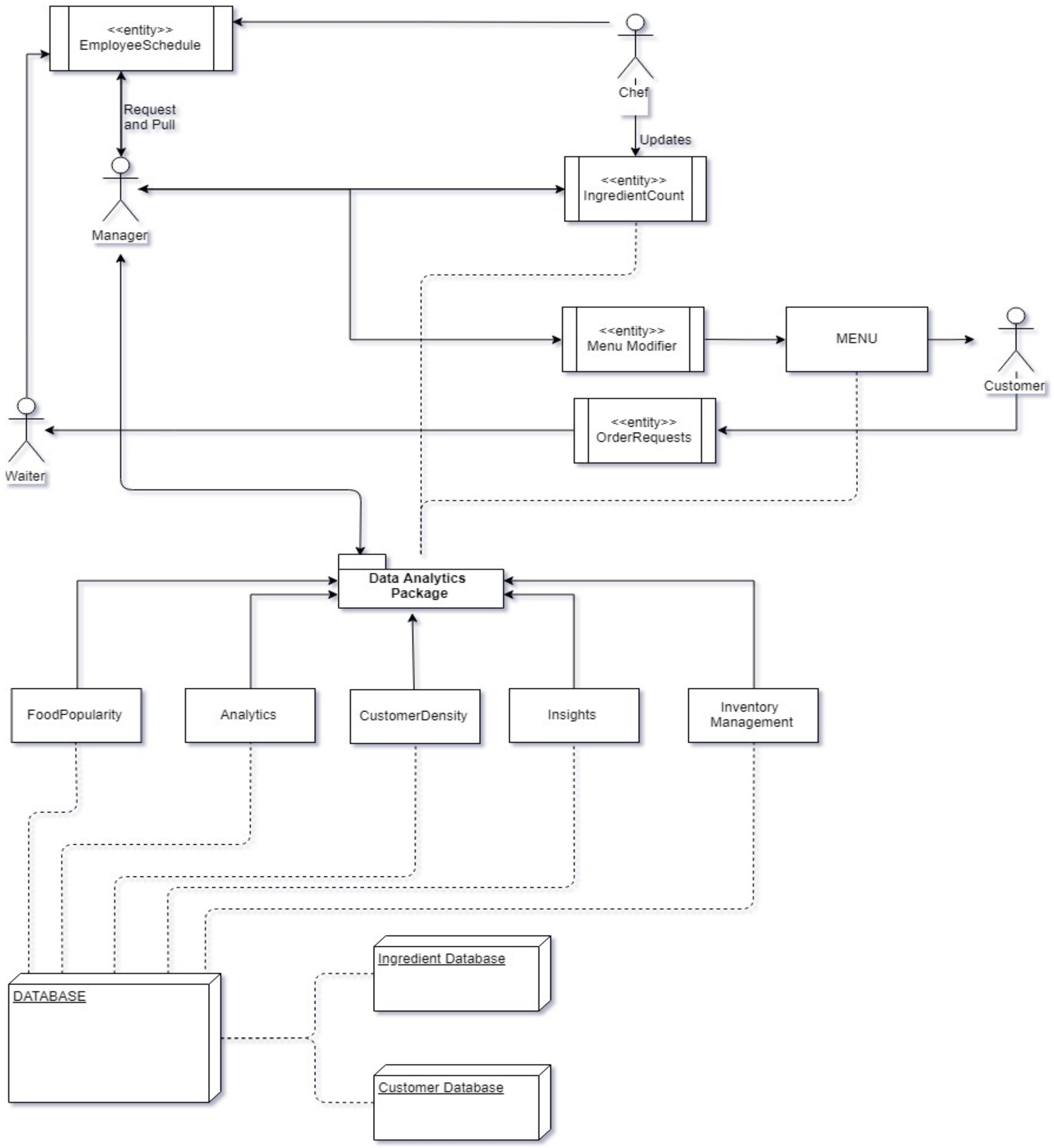


Figure 15: Domain model diagram

10.4 System Operations Contract

Table 21

Name	To place an order
Responsibilities	To take the order from the customer and select it in the FoodByte portal.
Use Cases	UC-1
Exceptions	None
Preconditions	- Menu Database should contain the updated values. - The FoodByte portal should be synced with the Menu Database.
Postconditions	The User Database should be updated with the Customer selection and notification

Table 22

Name	To create an artificial dataset
Responsibilities	To provide the dataset with all the parameters required for efficient working of the system
Use Cases	UC-2
Exceptions	None
Preconditions	The mathematical model to create the artificial dataset should be predetermined.
Postconditions	The User Database should be updated with the Customer selection and notification to be sent to the Data analytics modules to start processing.

Table 23

Name	Clustering of Data
Responsibilities	To retrieve entries from the User Database and cluster them into groups based on location and sub groups based on customer density, item sales and ethnicity.
Use Cases	UC-3
Exceptions	None
Preconditions	Users database should be updated with latest entries.
Postconditions	The system should provide Food Item vs Time, customer Density vs Time and Ethnicity data for further analysis.

Table 24

Name	Trend Analysis in sales based on time
Responsibilities	To retrieve entries from the Users Database and provide sales predictions at specific times

Use Cases	UC-4
Exceptions	None
Preconditions	- Users Database should updated with latest entries - The system should have the Food item vs time dataset in the required form
Postconditions	The system should use the Food item vs Time matrix from the dataset and provide predictions about the demand at specific times during the day

Table 25

Name	Analysis based on Customer density
Responsibilities	To retrieve entries from the Users Database and suggest menu updates based on type of customers
Use Cases	UC-5
Exceptions	None
Preconditions	- Users Database should updated with latest entries - The system should have the Food item vs Type of customer dataset
Postconditions	The system should use the Food item vs Type of customer matrix to predict peak working hours and suggest menu updates at those times

Table 26

Name	Trend Analysis based on Geography
Responsibilities	To retrieve entries from the Users Database and suggest price changes based on location
Use Cases	UC-6
Exceptions	None
Preconditions	- Users Database should updated with latest entries - The system should have the Food item vs Type of Location dataset
Postconditions	The system should use the Food item vs Location matrix and predict the price changes or special offers needed at specific locations of the restaurant

Table 27

Name	Inventory Management
Responsibilities	To retrieve entries from the Inventory Database and predict the required raw materials based on demand
Use Cases	UC-7
Exceptions	None
Preconditions	- Inventory Database should updated with latest entries - The system should have the Food item vs Inventory dataset
Postconditions	The system should use the Food item vs Inventory matrix, predict the rate
	of consumption of an ingredient and allow the restaurant to maintain the stock at peak times

Table 28

Name	Maintaining Statistics
Responsibilities	To create the statistical report based on the predictions of UC2 – UC5
Use Cases	UC-8
Exceptions	None
Preconditions	- UC2 – UC5 should have completed their predictions - The system should be able to generate a UI based report
Postconditions	The system should integrate the individual predictions of UC2 – UC5 into a single conclusion

10.5 Mathematical Models

1) Creating artificial datasets

Let N be the number of entries in the dataset. We will divide this number into P subsets. Each subset will have N/P entries. Each entry will contain values of Food type, Customer Density, Price, Location, Time of the day and ethnicity. We will use **Gaussian distribution** between the P subsets to calculate the probability of the value for each field within that subset.

$$f(x \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

where

- μ is the mean or expectation of the distribution (and also its median and mode),
- σ is the standard deviation, and
- σ^2 is the variance.

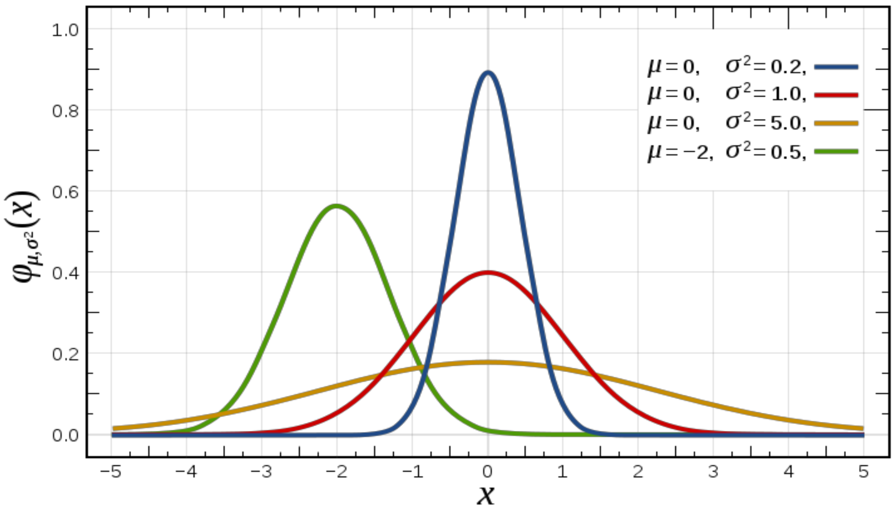


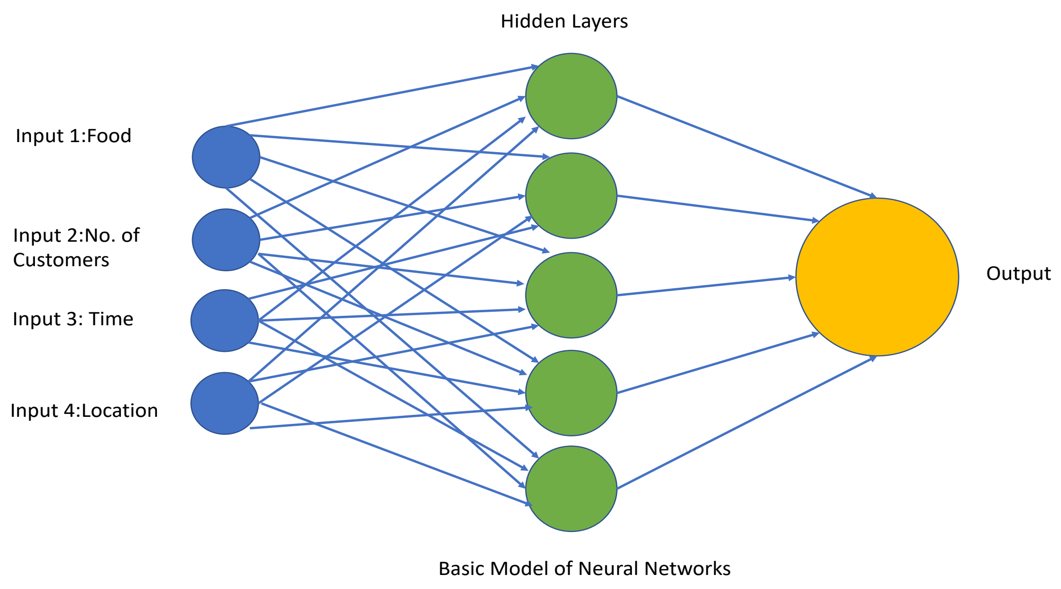
Figure 16

For example, x% of people in southern part of the US prefer Mexican food (Initial data from Statistical report). This value will be considered as mean and will be part of the 1st subset. The rest P-1 subsets will have values ranging from (x-sigma)% to (x+sigma)%. This method will be used for all the food categories, locations and time values. This will create a dataset which will be pseudo random, i.e initial data based on observations and variation based on mathematical modelling.

2) Classifiers

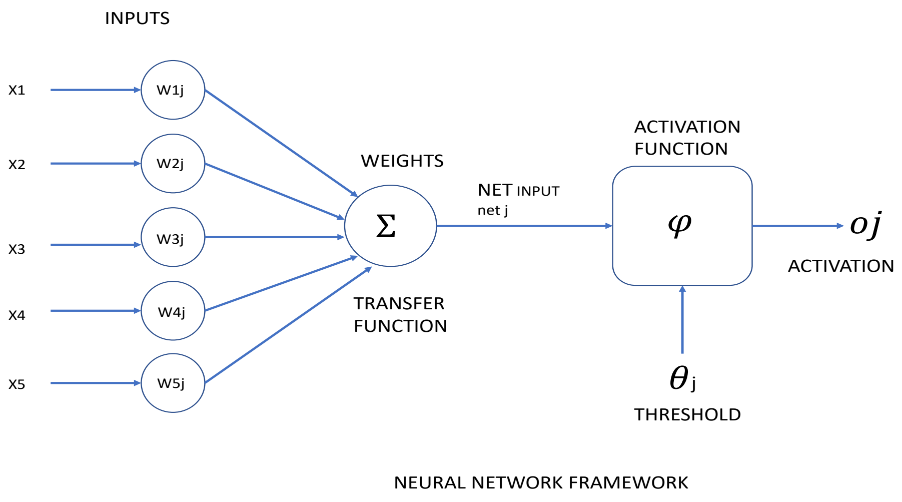
Our project uses the collection of datasets as the input and uses machine learning tasks to perform computations that yield meaningful results relevant to the outlined objectives of the project. We will use the supervised learning approach to perform analysis on the data. Specifically, classifiers will be used to observe the input data and then classify into new observations. We have chosen the structural classifier neural networks computing system to manipulate our data.

Figure 17: Basic model of neural networks



As the figure above shows, we feed our collected data as inputs and pass them through a hidden layer to perform some manipulations through which we obtain an output that can be interpreted.

Figure 18: Neural network framework



In the diagram above, we choose the 6 inputs to be the type of food, its price, the location of the restaurant, the number of customers and the time that the order was placed. The inputs are then multiplied with weights and combined to form the net input which is then sent to the activation function, which adds non-linear properties to the data. Depending on the accuracy and relevance of the output rendered, the kind of activation function will be chosen from the Python libraries.

11. INTERACTION DIAGRAMS

UC 1- To place the order

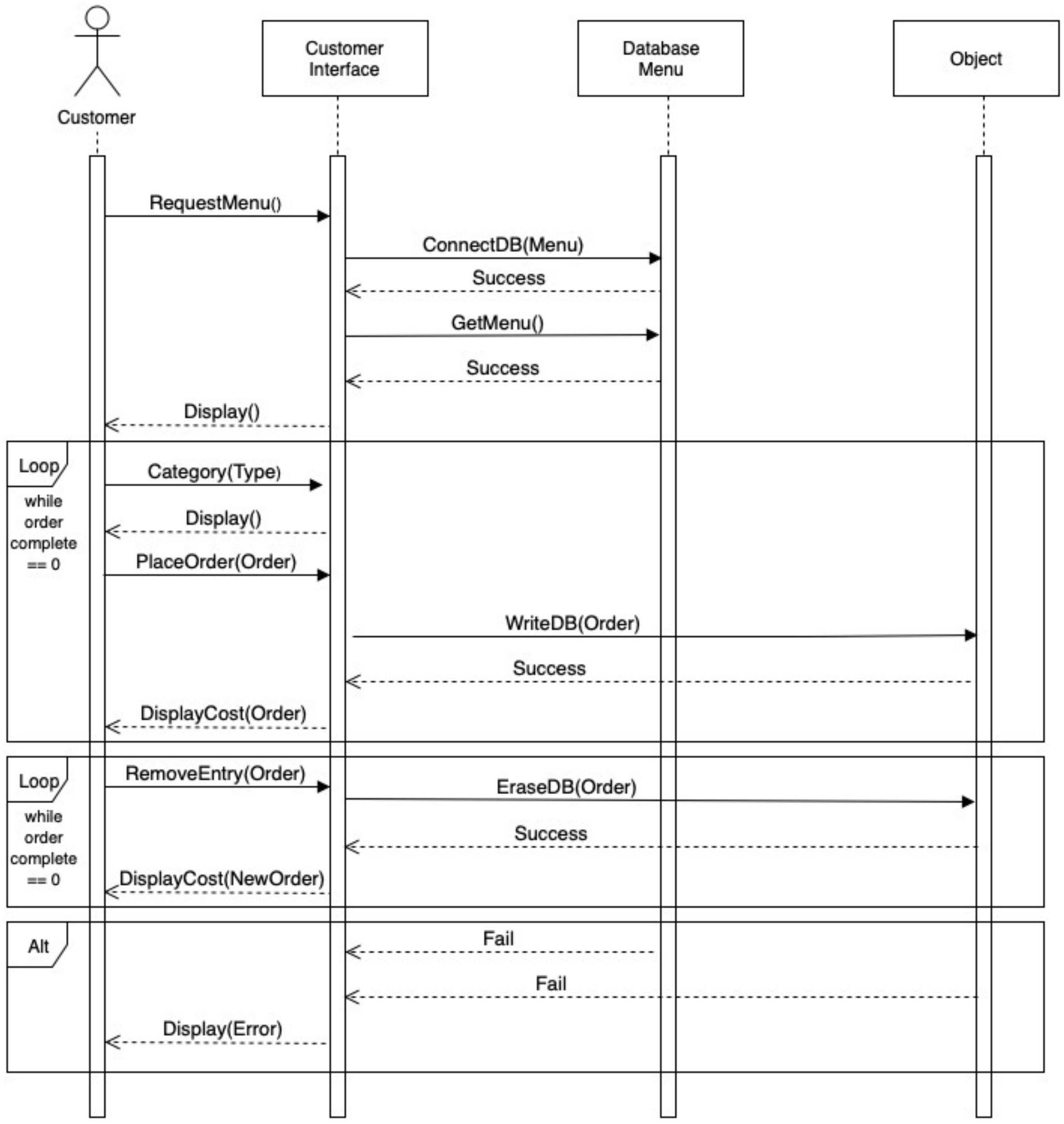


Figure 19: To place an order

The customer requests for the menu and it is retrieved from the database. Next, based on the category of food item selected the particular sub menu is presented. The customer has an option to select items from the menu which is then updated in the Users database. The individual costs of the item and the total bill is presented to the customer after which he/she can modify the items selected as per their convenience. If modified, the Users database is updated, and the new total cost is presented to the customer.

UC2- Creating Artificial Dataset

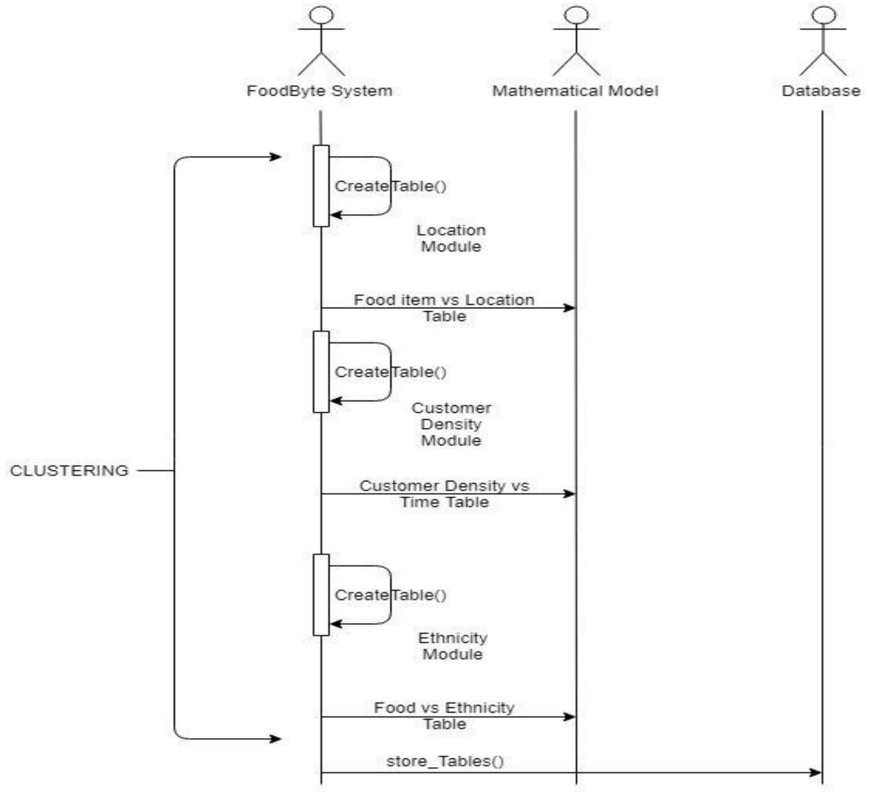


Figure 20: Creating artificial dataset

The artificial dataset creation is based on a mathematical model which uses the Gaussian Probability Distribution function. Based on the data available, the dataset for Food vs Location, CustomerDensity vs Time and Food vs Ethnicity will be created. The variation in parameters will follow normal distribution, thereby having a realistic dataset that is needed for analysis.

UC3- Clustering

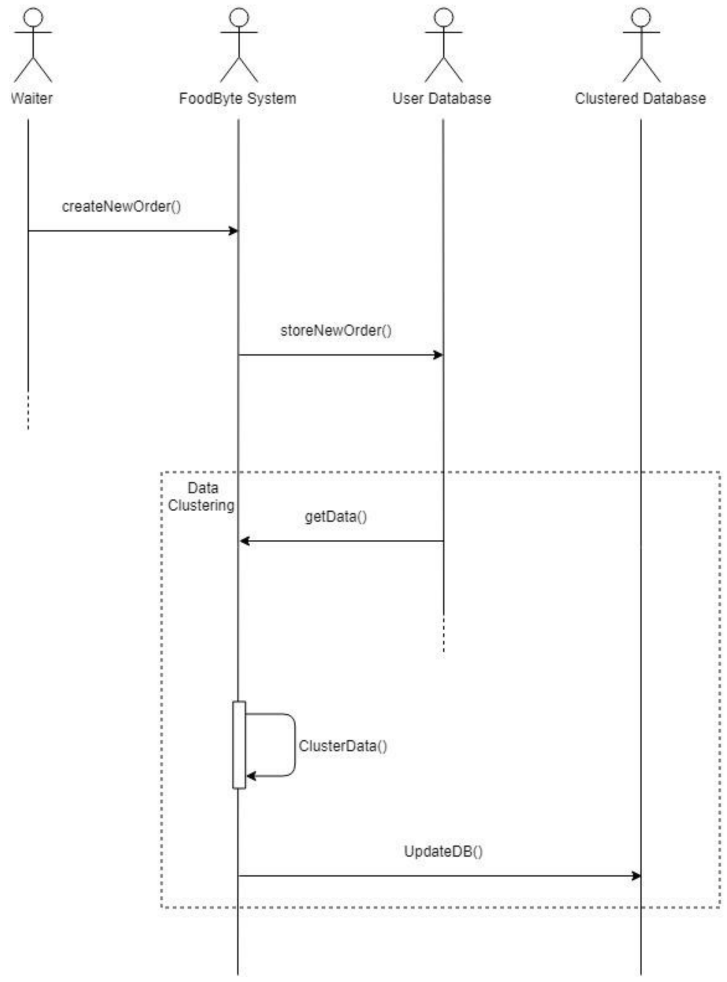


Figure 21: Clustering

Clustering module will read the data from the Database and organize the entries based on Ethnicity and Location to find the most popular food item within each zone.
The dataset can be considered as distributed over x months, to find the decrease or increase in popularity of each food item. This can be used to further update the Menu to add or delete items or update the Price.

UC4- Analyze trend in sales based on time

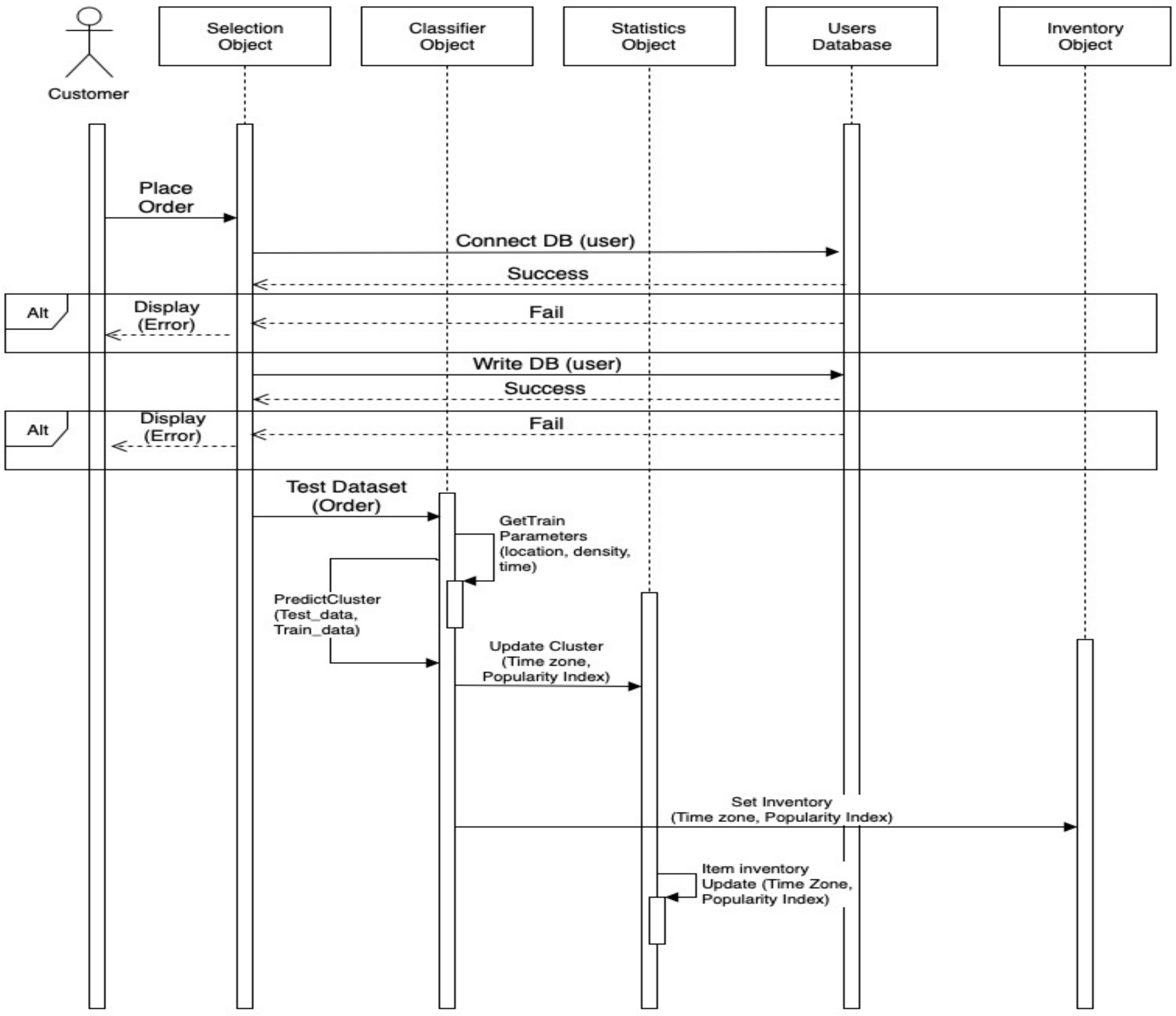

Customer
Selection Object
Classifier Object
Statistics Object
Users Database
Inventory Object
Place Order
Connect DB (user)
Success
Alt
Display (Error)
Fail
Write DB (user)
Success
Alt
Display (Error)
Fail
Test Dataset (Order)
GetTrain Parameters (location, density, time)
PredictCluster (Test_data, Train_data)
Update Cluster (Time zone, Popularity Index)
Set Inventory (Time zone, Popularity Index)
Item inventory Update (Time Zone, Popularity Index)

Figure 22: Trend in sales based on time

The customer places an order through the customer interface, which then gets stored in the Users Database. The classifier object receives the entry as a test input for Location. It will use the cached training dataset parameters and predict the location cluster based on the training parameters. This new entry will then be saved to the Location cluster in the Statistics object and the Inventory object. The system will then check if there are menu updates or price updates are necessary and save this result. The menu or price update analysis will happen periodically in the background.

UC5-Analysis based on customer density

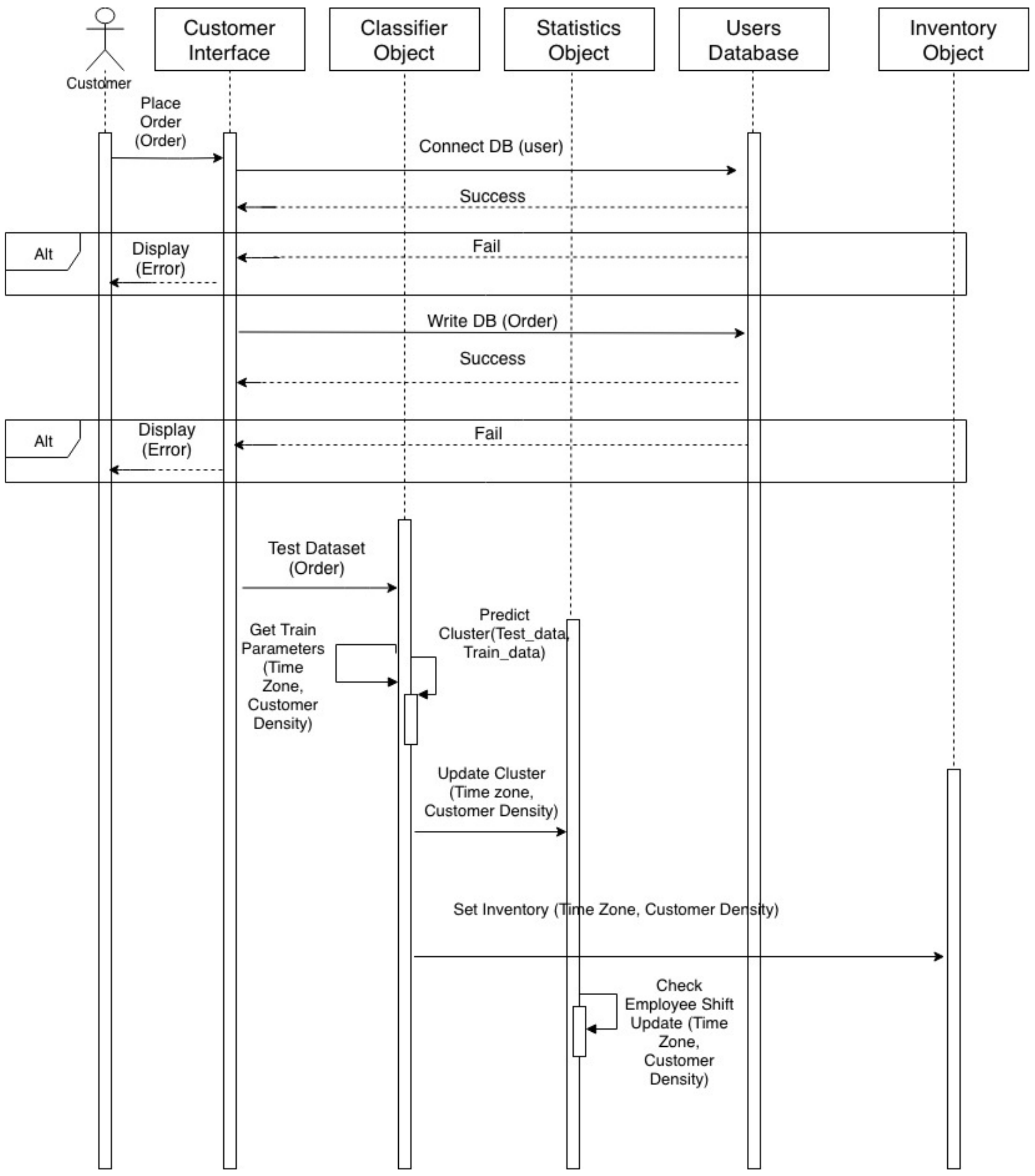

Customer
Customer Interface
Classifier Object
Statistics Object
Users Database
Inventory Object
Place Order (Order)
Connect DB (user)
Success
Alt
Display (Error)
Fail
Write DB (Order)
Success
Alt
Display (Error)
Fail
Test Dataset (Order)
Get Train Parameters (Time Zone, Customer Density)
Predict Cluster(Test_data, Train_data)
Update Cluster (Time zone, Customer Density)
Set Inventory (Time Zone, Customer Density)
Check Employee Shift Update (Time Zone, Customer Density)

Figure 23: Analysis based on customer density

The customer places an order through the customer interface, which then gets stored in the Users Database. The classifier object receives the entry as a test input for Customer Density. It will use the cached training dataset parameters and predict the Customer Density cluster based on the training parameters. This new entry will then be saved to the Customer Density cluster in the Statistics object and the Inventory object. The system will then check if any employee shift updates are necessary and save this result. The Employee shift update analysis will happen periodically in the background.

UC6-Trend analysis based on geography

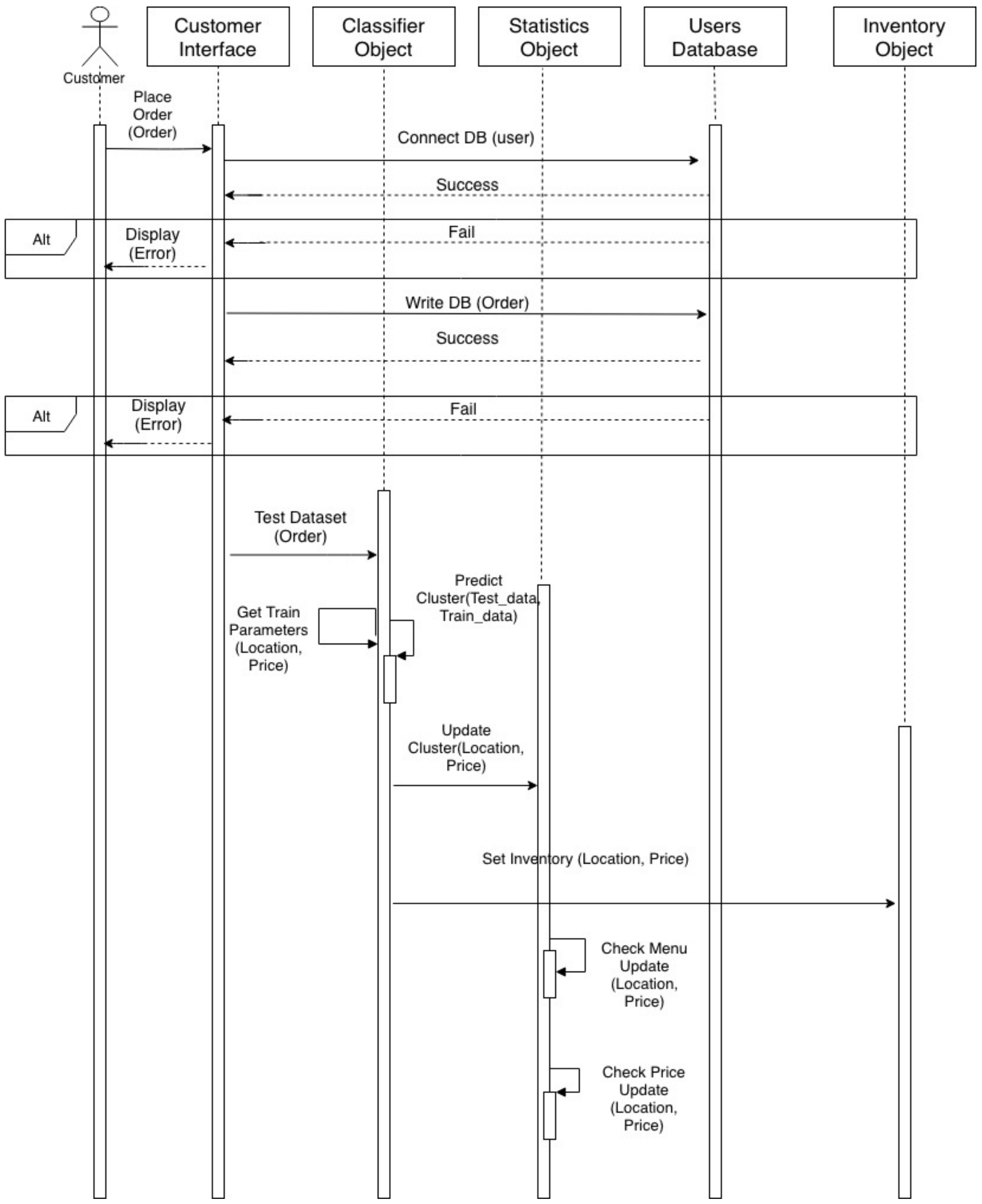


Figure 24: Trend analysis based on geography

The customer will place the order using the customer interface. This entry will be stored in the Users Database. The classifier object will receive this as test input for location. It will use the cached training dataset parameters. The location cluster will be predicted based on the training parameters and the input location. This new entry will be saved to the predicted location cluster in the statistics object and the Inventory object. The system will then check if any Menu updates or Price updates are necessary and save this result. The Menu and Price update analysis will happen periodically in the background.

UC7- Inventory management

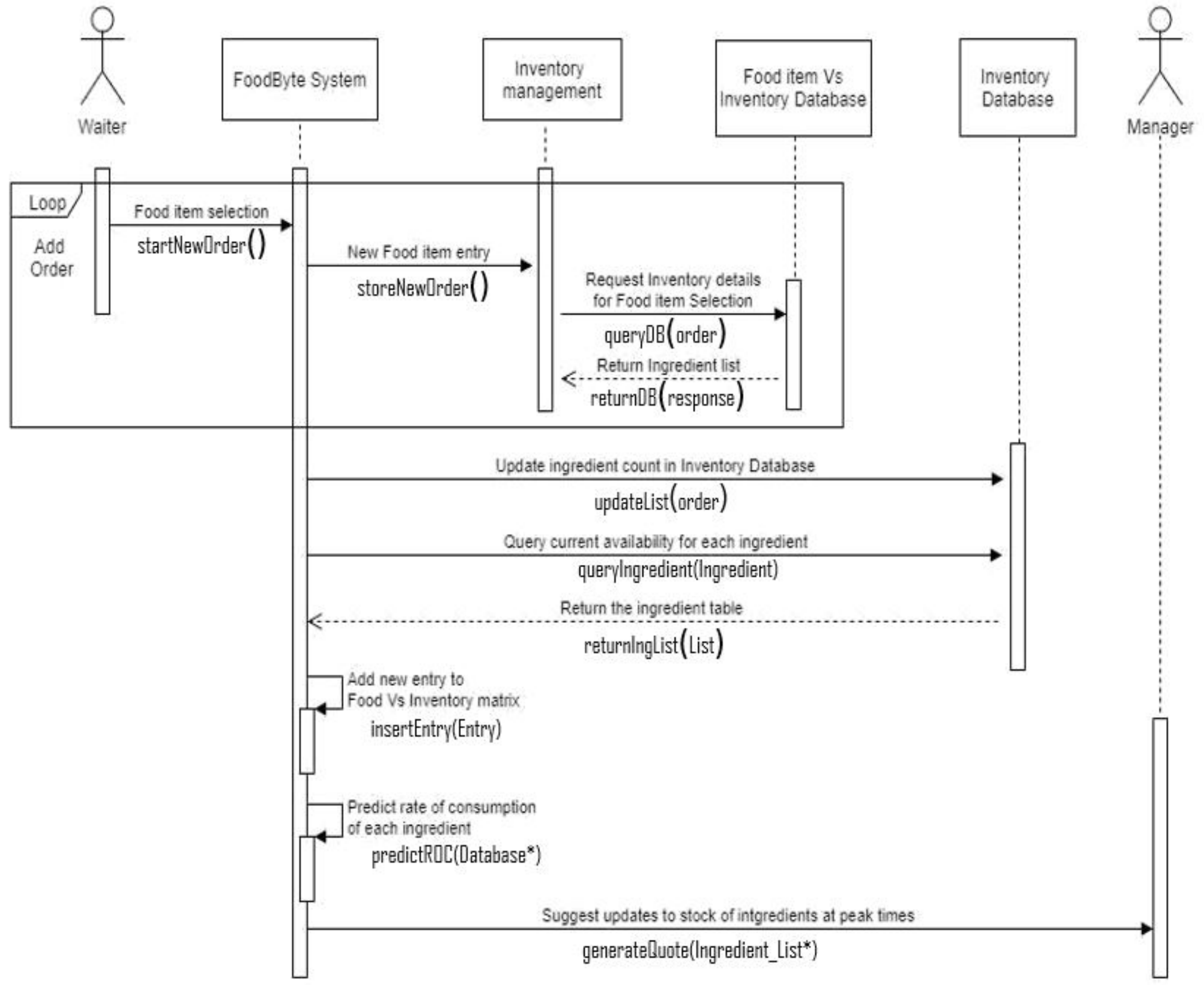

Waiter
FoodByte System
Inventory management
Food item Vs Inventory Database
Inventory Database
Manager
Loop
Add Order
Food item selection
startNewOrder()
New Food item entry
storeNewOrder()
Request Inventory details for Food item Selection
queryDB(order)
Return Ingredient list
returnDB(response)
Update ingredient count in Inventory Database
updateList(order)
Query current availability for each ingredient
queryIngredient(Ingredient)
Return the ingredient table
returnIngList(List)
Add new entry to Food Vs Inventory matrix
insertEntry(Entry)
Predict rate of consumption of each ingredient
predictROC(Database*)
Suggest updates to stock of intgredients at peak times
generateQuote(Ingredient_List*)

Figure 25: Inventory management

In this case, the order received by the waiter is entered into the FoodByte Portal. The system then enters the new order into the Inventory Management System. The Inventory Management System queries the list consisting food item vs ingredients to know the exact ingredients used in the dish. This response is then used to update the Ingredient Database. The updated Ingredients list is then returned to Inventory Management System. In case a new Food Item is added to the System, the Food vs Inventory list is updated by the FoodByte System. To predict rate of consumption of each ingredient, a database pointer is used, and the prediction algorithms return the Rate of Consumption of each Ingredient. When the manager requests a quote, the Ingredient List is used along with the data analytics algorithms to generate a new quote.

UC8 – Maintaining Statistics

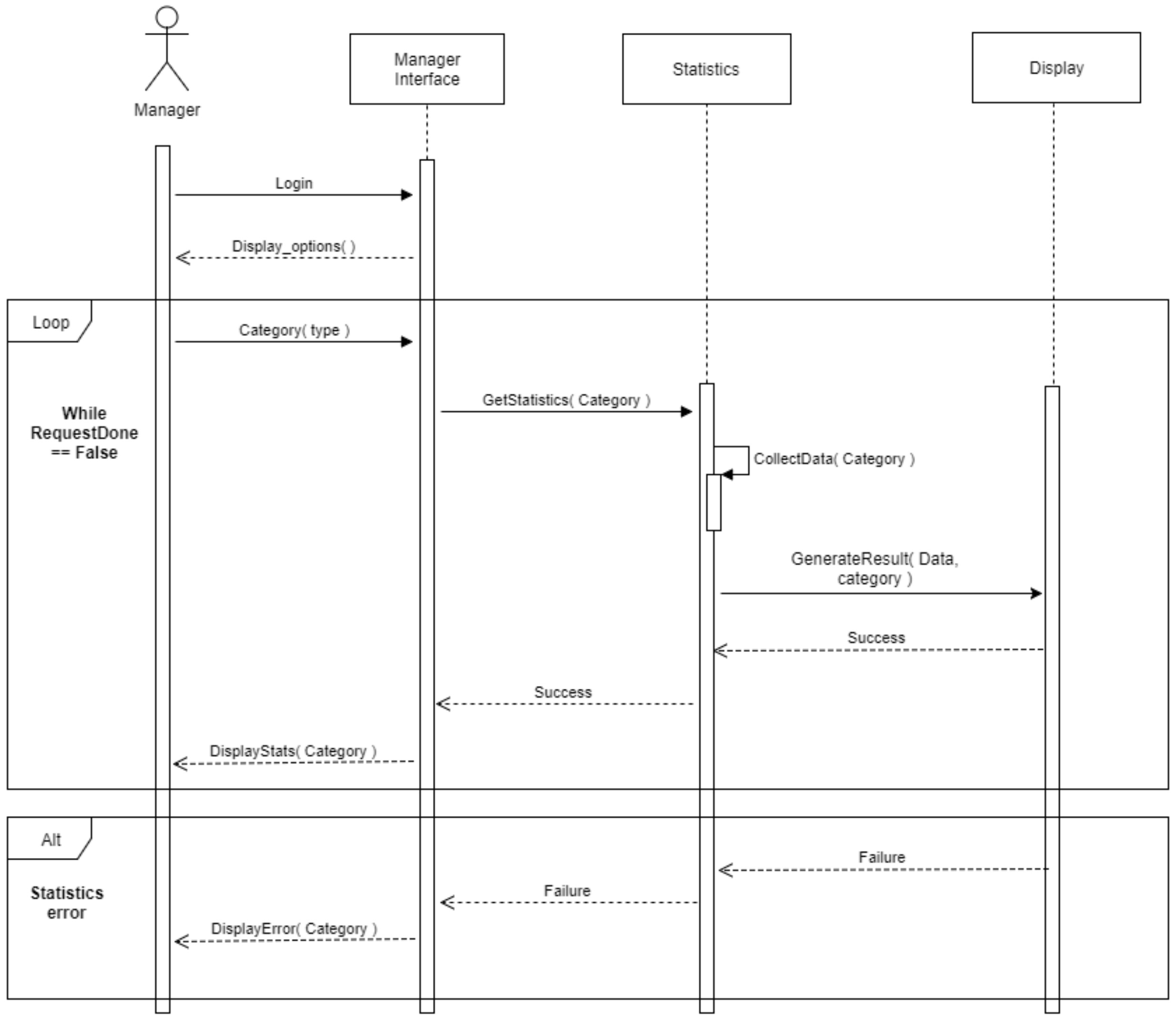


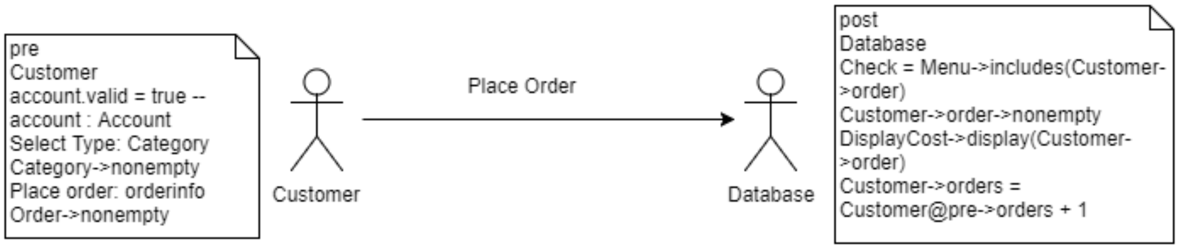
Figure 26: Maintaining statistics

The Manager will login to through the interface and view the statistics options. The Manager will select a specific category for example, Location Data, Hourly Customer Pattern or Employee requirement from the options. The Statistics module will receive this request and collect the data for this category. It will then provide this data pointer to the display module which will convert it into a suitable display format like bar graph, histogram or Pie chart. The result will then be displayed on the interface for the manager. In case of any errors, an error message will be displayed.

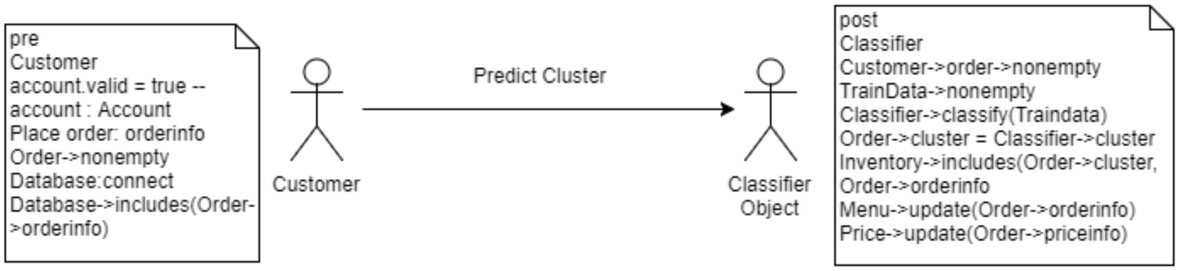
12. OBJECT CONSTRAINT LANGUAGE (OCL) CONTRACTS:

- Contract invariants, pre and post conditions

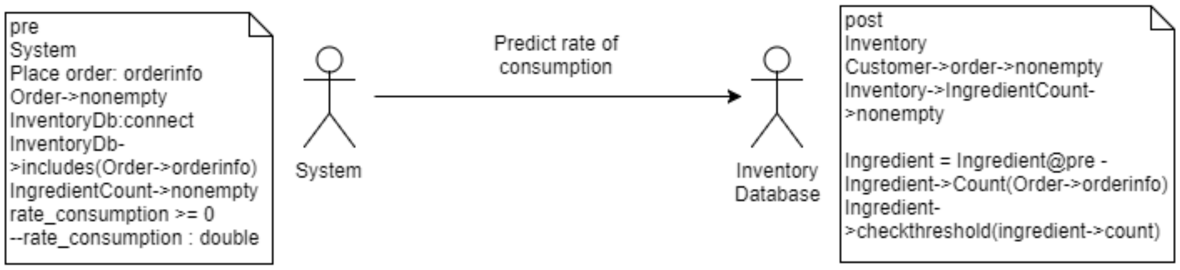
1) UC1 – Place Order



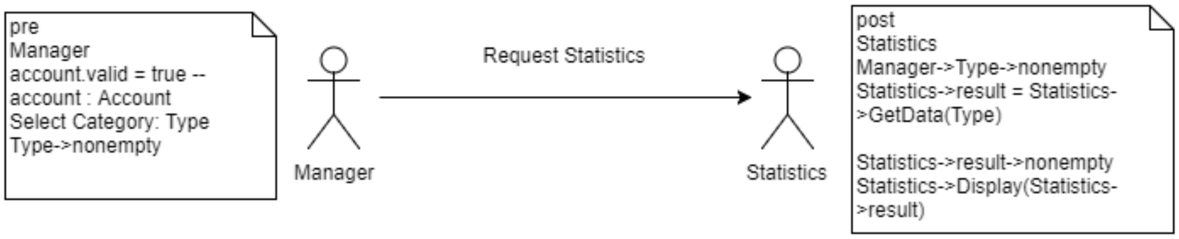
2) UC2, UC3, UC4 – Trend Analysis


pre
Customer
account.valid = true --
account : Account
Place order: orderinfo
Order->nonempty
Database:connect
Database->includes(Order->orderinfo)
Customer
Predict Cluster
Classifier Object
post
Classifier
Customer->order->nonempty
TrainData->nonempty
Classifier->classify(Traindata)
Order->cluster = Classifier->cluster
Inventory->includes(Order->cluster, Order->orderinfo
Menu->update(Order->orderinfo)
Price->update(Order->priceinfo)

3) UC5 – Inventory Management


pre
System
Place order: orderinfo
Order->nonempty
InventoryDb:connect
InventoryDb->includes(Order->orderinfo)
IngredientCount->nonempty
rate_consumption >= 0
--rate_consumption : double
System
Predict rate of consumption
Inventory Database
post
Inventory
Customer->order->nonempty
Inventory->IngredientCount->nonempty
Ingredient = Ingredient@pre - Ingredient->Count(Order->orderinfo)
Ingredient->checkthreshold(ingredient->count)

4) UC6 – Maintaining Statistics



13.CLASS DIAGRAM AND INTERFACE SPECIFICATION

13.1.1 Class Diagram

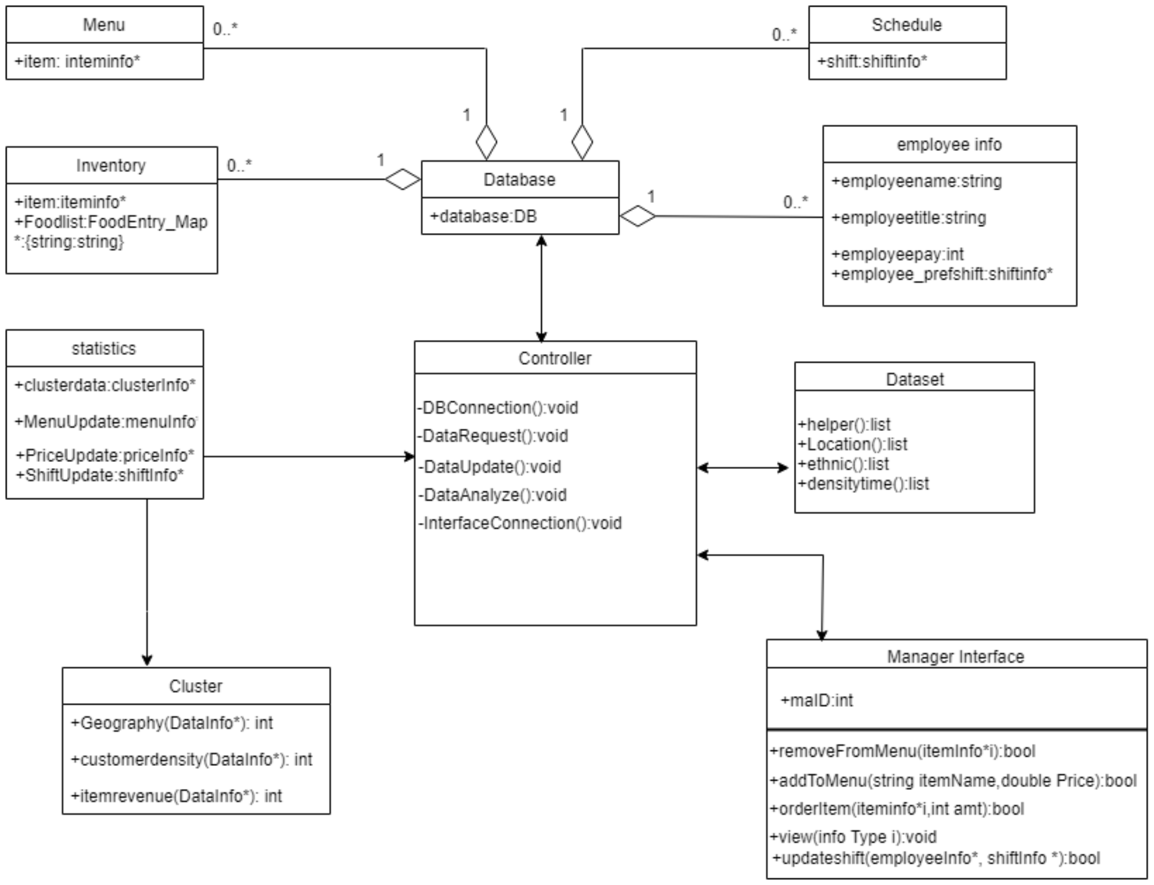


Figure 27: Class diagram

13.1.2 Design pattern:

Our system uses the command design pattern. Whenever a request is generated it is encapsulated in the form of object thereby letting us parameterize clients with different requests. For example, when the user wants to view the analysis based on geography, the object for this request is created and the user can view the results. Similarly, for all the other types of requests their respective objects are created to handle the results. Another reason to use the command design pattern is that many times the command can be undone. In a case where the waiter makes some mistake in taking order from the customer then it can be undone easily.

13.1.3 Class Descriptions

Menu: Menu is a class which contains all the menu items and each item's corresponding information.

Schedule: Schedule is a class that represents the timetable of all employees working during a given time. Upon calling the function 'shiftinfo' within the Schedule class, it will display how many employees are working each shift and the corresponding number of customers at that shift time.

Inventory: Inventory is a class that contains the quantity of items used in orders. Upon calling the function 'Iteminfo' within the Inventory class, it will display the current quantity of all items and which items are running low.

Employee Info: Employee Info is a class which contains information about each employee including the name, position, and pay rate.

Database: Database is a class which stores all information about FoodByte including the schedule, employee information, inventory, business statistics, menu information, etc.

Statistics: Statistics is a class which contains functions that allow the Manager to view profits and expenses either quarterly or yearly, and the results of analysis based on Geography, Customer Density, Popularity of items, and Inventory.

Controller: The Controller requests, updates, and shares information between the Database and the Dataset, the Cluster, and the Manager Interface.

Dataset: The Dataset contains individual lists of data of the location, Ethnicity, Food type, Customer Density, Time, etc.

Cluster: Cluster is used to group data for analysis such as, Geography Vs. Food Type, Customer Density Vs. Time, Popularity of items Vs. Time, etc.

Manager Interface: The manager interface is used to carry out essential tasks to the managing of the restaurant. Through it, the manager can manage shifts of employees, change wages of employees, track revenue and operating costs, alter the menu, manage Inventory, etc.

13.2 Data Types and Operation Signatures

Menu

The Menu class will have the food category, food item and price list. This class will handle the Menu database query and updates.

Attributes

+item:iteminfo*	List containing Category, Food item and price columns

Methods

+UpdateMenu(Item,action):bool	Method gets the item and action from Manager's Interface and sends entry to the Database class to update the table. Returns true if the entry is successfully updated.

Inventory

The Inventory class has the maintains the Food item vs Ingredient and Ingredient count lists. The Ingredient count list is updated based on the new Food item entry into the Users Database

Attributes

+Foodlist:FoodEntry_Map*:{string:string}	Hash map of Ingredients with Key as the Food Entry
+IngredientList:Ingredient_info*	Quantity of each ingredient in the Ingredient list
+ExpiryList: expiry_info*	Maintains the list of remaining number of days before the expiry of each ingredient

Methods

+UpdateIngredient(Ingredient, FoodEntry, Action):bool	Adds/Removes Ingredient from the ingredient list for input FoodEntry based on value of Action
+UpdateAvailableCount(Ingredient):int	Updates available count of Ingredient in the Database
CheckRefill(Ingredient,count):bool	Returns true if the ingredient count is below the recommended threshold
GetRefillStatistics():list< Ingredient_list*>	Returns a list of all the ingredients that needed refill. Value is used by the statistics module.
GetExpiryList(Ingredient_list *): list<expiry_info*>	Get the expiry period of all the items in the Ingredient list

Schedule

Schedule class maintains a list of the shifts of all the employees in the restaurant. This list is needed by the prediction module to check for shift and wage adjustments of the employees

Attributes

+shift:shiftinfo*	List of shifts for each employee

Methods

GetShiftInfo(EmployeeID, shiftInfo *)	Returns the shift information for the given employee ID
+ChangeShift(EmployeeID, shiftInfo)	Updates shift of the employee and sends it to database

Employee Info

This class maintains the Employee ID, name, Title and preferred shift data. This data will be used by the prediction module to manage the number of employees during a particular shift.

Attributes

employeeInfo * - contains following data types

+employee_ID: int	Employee ID
+employee_name:string	Employee Name
+employee_title:string	Employee Title
+employee_prefshift:shiftinfo*	Employee Preferred shift information

Methods

+AddEmployee(name,ID,Title,shiftInfo*):bool	Add new employee information into the database
+RemoveEmployee(ID):bool	Remove an employee from database using employee ID
+UpdateShift(ID,shiftInfo*):bool	Updates the preferred shift information for the employee
GetShift(ID):shiftInfo*	Get the preferred shift for employee

Cluster

This class contains the clustering algorithms to classify the data based on Location, Ethnicity and Time Zone. This data will be used for prediction of cost, menu updates and employee shift adjustments.

Attributes

+clusterIndex:clusterInfo*	Contains the cluster Information of each Entry
+predresult:predictInfo*	Contains the prediction result of each cluster Along with the associated parameter like menu, cost or shift

Methods

All the clustering methods will use the clustering algorithm to converge the datapoints into a single cluster. The Data points will then be grouped into that cluster for statistical analysis.

Geography(DataInfo *): int	Returns the cluster index of the DataInfo set based on FoodType vs Location clustering
customerdensity(DataInfo *): int	Returns the cluster index of the DataInfo set based on time vs customer density clustering
itemrevenue(DataInfo *): int	Returns the cluster index of the DataInfo set based on time vs FoodType clustering

Statistics

This class will use the data points grouped in each cluster, generate statistical report(Table) based on this data. It will also suggest menu updates, price updates and employee shift updates if deducted from the clustering pattern

Attributes

+clusterdata:clusterInfo *	A list which contains detailed information of each cluster identified i.e num_entries, convergence rate, fixed parameter, variable parameters
+MenuUpdate:menuInfo *	A list of all items which need menu update and the update characteristic (Add/Delete/Modify)
+PriceUpdate:priceInfo *	A list of all items which need price update and the update characteristic (Increase/Decrease) and the range of change
+ShiftUpdate:shiftInfo *	A list of all employees (ID) who’s shift needs to be updated and the updated shift information.

Methods

ShiftAnalysis(clusterInfo *,shiftInfo *,predictInfo *):bool	Performs shift analysis on the input cluster and updates the prediction result based on the preferred shift of each employee. Returns true if the analysis predicts any updates
RevenueAnalysis(clusterInfo *, predictInfo *):bool	Use the cluserInfo to check for price variation over a period of time and update the prediction result for price updates if needed
+UpdateLists(Listtype, clusterInfo *):void	Based on the listtype, updates the menu list, price list or shift list in the input cluster

Manager Interface

This class gets the actions as input from the manager. These actions will be to update menu, price or employee shifts. It will also display the statistical results and suggestions on the interface when requested.

Attributes

+ManagerID: int	Manager's ID needed for login authentication

Methods

View(InfoType *):void	View statistical data based on the selection type from the interface
+addToMenu(itemInfo *,Price):bool	Add new item to Menu Database
+removeFromMenu(iteminfo *):bool	Remove and item from the Menu database
+updatePrice(itemInfo *,NewPrice):bool	Change the price of a food item
+updateshift(employeeInfo *, shiftInfo *):bool	Change the shift of an employee.

Dataset

This class generates the artificial dataset based on a mathematical model. It has 3 sub-classes for Location, Customer Density, Ethnicity for each type of dataset variation and a helper sub-class for data sharing.

Sub-class names – Location, DensityTime, EthnicState, Helper

Attributes

Datatype hashmap - {Key:Value}

+FoodType:hashmap {string:string*}	Food category as Key and List of Food entries in that category as value
+food_data:hasmap{int:string}	Numerical entry as Key for each Food Category as value
+zone_data:hashmap{int:string}	Numeric key for each location zone(North,South,East,West,Central) as value
+num_entries:int	Number of entries to be generated and stored in the user database
+food_entry:hashmap{int:string}	Numeric Key for each Food item as a value
+price:hashmap{int:string}	Price as a Key for each Food item as a value
+food_type:hashmap{string:int*}	Breakfast, Lunch, Snacks, Dinner as Keys for corresponding time zones for each as values

+customer_per_time:hashmap{string:int}	Percentage of customers as values for each food type(food_type) as Key
+ethnic_data:hashmap{int:string}	Numeric key for ethnicity type as values

Methods

+__init__(self): void	Constructor. Initializes all the Data structures and variables for each sub-class
+CreateLocationDataset(self): list<char *>	Creates the location vs FoodEntry dataset and return the count of each food entry for each location in the table
+CreateDensityTime(self): list<char *>	Creates the customer density vs Time dataset and returns the count of each Density value for each time in the table
+CreateEthnicset(self): list<char *>	Creates the Ethnicity vs Location dataset and returns the count of each ethnicity entry for each location value in the table
GetKeyByValue(self,dictionary,value):int	Computes the Key for a single or array of values in the input dictionary(hashmap)
GetFoodCategory(self,FoodEntry):String	Returns the food category for the input Food Entry
RoundingCorrection(self,array,percentage) :void	Removes the error generated for the entry values during floating point rounding off
GetRandomizationRange(self,FoodCount, EthnicCount,FoodEntry): list<char *>	Computes the optimum random range of zone key and ethnic key values that can be used for creating the dataset

Controller

This class handles the database connection requests based on database identifier and executes the INSERT, CREATE, DELETE queries on the database table specified by the input values.

Methods

+ConnectDB(ID):bool	Establish connection with the database
+RequestEntry(requestInfo *):void	Requests the data entry or set of entries from the table based on the request info input
+UpdateEntry(updateInfo *):void	Update an entry in the database based on update info input
+DataAnalyze():void	Method to read and save the analysed statistics in the correct format into the database
+InterfaceConnection():void	Method to connect to the Manager's portal and check for new actions needed

Database

Attributes

+Database:DatabaseID:int	Database Identifier
+Database:Table: string	The table name in that database
+EmployeeInfo:Employee_ID: int	Employee ID
+EmployeeInfo:Employee_name: string	Employee name
+EmployeeInfo:Employee_Title: string	Employee Title
+EmployeeInfo:Employee_shift: string	Preferred shift of the employee
+Menu:FoodEntry: string	Food Entry
+Menu:FoodCategory: string	Food Category
+statistics:revenue: double	Revenue of the restaurant
+Inventory:Ingredient: list<itemInfo *>	List of ingredients for Food Entry
+Inventory:IngredientCount: int	Ingredient count available
+Schedule:ShiftInfo:shiftInfo *	Shift details per employee

13.3 Traceability Matrix

Table 21: Traceability Matrix

	Software Classes									
Domain Concepts	Menu	Inventory	Statistics	Schedule	Information	Interface	Cluster	Database	Controller	Creation
Manager	x	x	x	x	x	x				
Waiter	x			x						
FoodPopularity			x			x	x			
OrderQueue								x		
Location	x	x	x	x	x	x	x	x		x
Order		x						x		x
Schedule			x	x	x	x		x	x	
IngredientCount		x	x			x		x		

MenuModifier	x		x			x			x	
CustomerDensity			x	x		x	x			x
DatabaseConnection	x	x	x	x	x			x	x	x

14. SYSTEM ARCHITECTURE AND SYSTEM DESIGN

14.1 Architectural Styles

FoodByte makes use of multiple Software Architectural styles to make it a robust and ubiquitous software solution. Majorly since our project is based on Data, the Data Centered Architecture is used. Part of our project also utilizes the Client Server architecture. Components like User Authentication and Manager request for insights are based on Request Response based communications.

The use of these different architectures is elaborated further.

Data-Centered Architecture

Our system is solely dependent on a central database, consisting of multiple tables. This database is accessed frequently by different components, which may/ may not modify the data within.

Components like Inventory Management System, various Trend Analysis Modules and all other Data Analytics modules frequently access the database to generate a response to the queries made through the portals. Thus, this database can be treated as a Shared Repository. This helps the system achieve integrality of data. Most of our components are independent of each other. The only intermediary of communication between them is the Central Database.

This makes our system have two major components:

a. *Central Database* - A structure or data repository, which is responsible for providing a permanent storage of data. The current state can be accessed using the Central Database.
b. *Data Accessors* - A group of independent components that operate on the Central Database, run algorithms using the data, and publish the results to the user.

The following figure illustrates the Data Centered Architecture as discussed above.

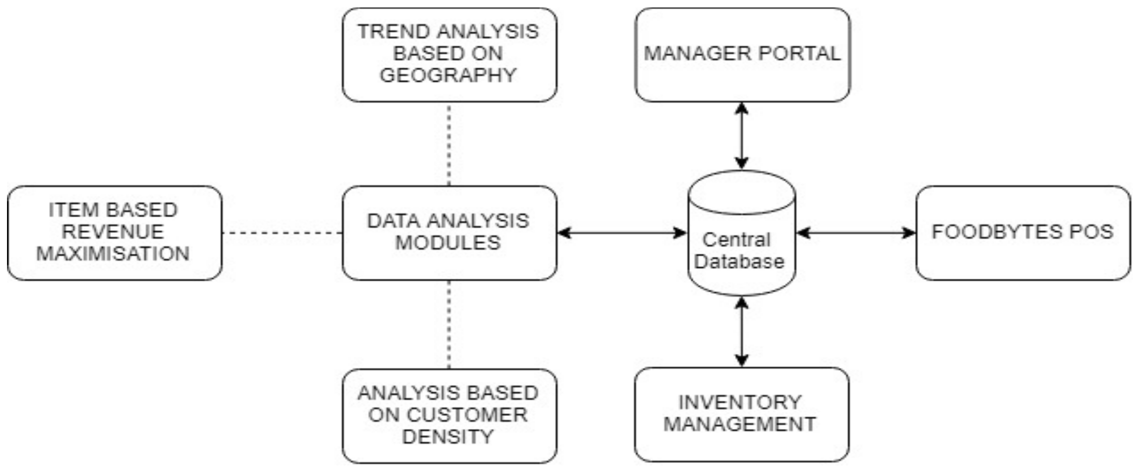

TREND ANALYSIS BASED ON GEOGRAPHY
MANAGER PORTAL
ITEM BASED REVENUE MAXIMISATION
DATA ANALYSIS MODULES
Central Database
FOODBYTES POS
ANALYSIS BASED ON CUSTOMER DENSITY
INVENTORY MANAGEMENT

Figure 28: Data Centered Architecture implemented by FoodByte

Client-Server Architecture

Part of our system also exhibits the behavior of the Client Server based architecture. The FoodByte POS (Point of Sale) system interacts with a server in order to receive information. For example, when an order is placed on the POS, a request is sent to the Inventory Management System (with the order as an argument), which checks the availability of the ingredients and confirms availability as a response.

The FoodByte POS(Client) portal logins take place on a client server-based architecture. The client requests for access with the login and password. The server, upon receiving this response, checks the credentials and either grants or denies access to the client, as a response.

Another example is when the Manager (Client) requests for insights based on data, the Portal sends requests to the Server (Data Analysis Module). In response to this, the Data Analysis Modules collect and compute results which are then published on the Managers Portal.

The following figure illustrates the Data Centered Architecture as discussed above.

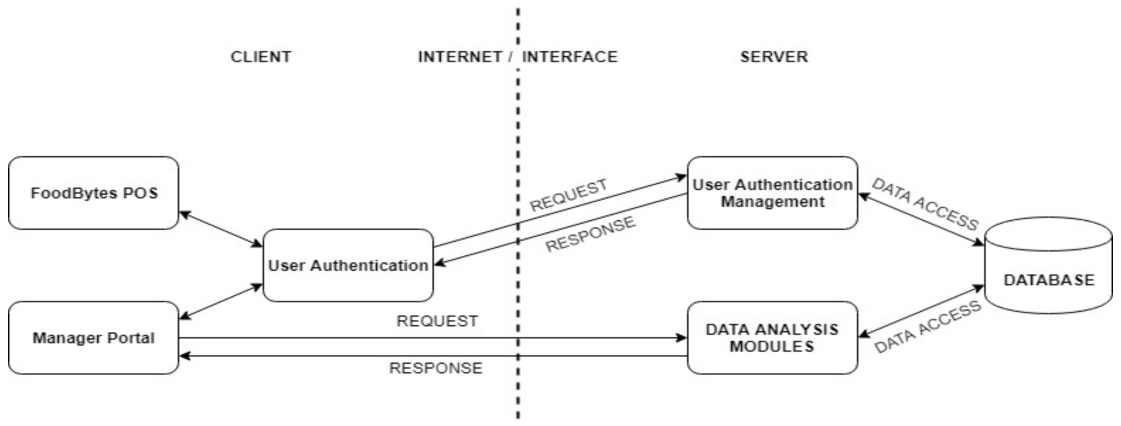


Figure 29: Client Server Architecture implemented by FoodByte

14.2 Identifying Subsystems

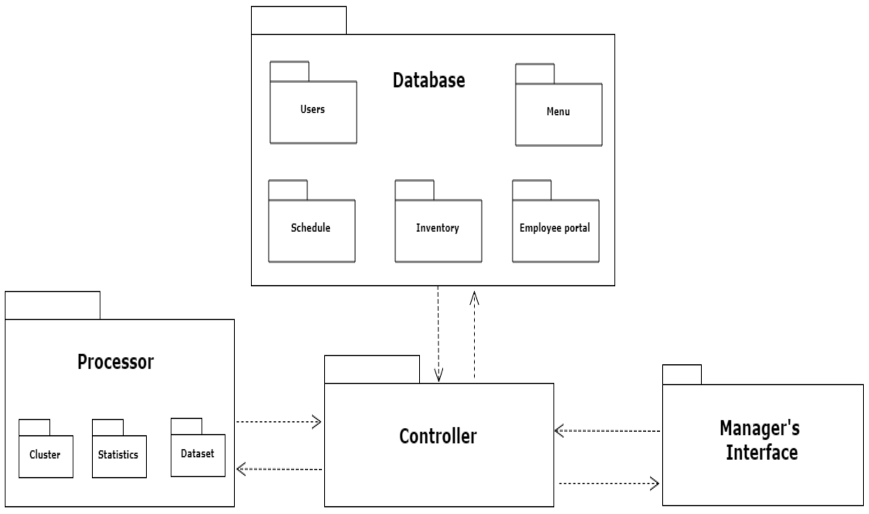


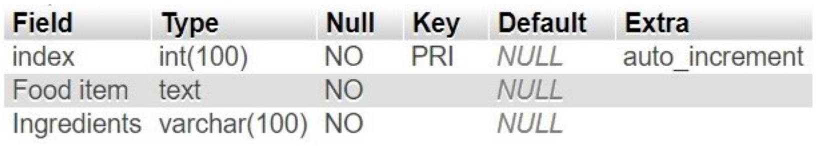
Figure 30: Subsystems

The above diagram describes the identification of the subsystem using a packaging diagram. The packages are split into 4 major components. The manager interface is used by the manager to check business statistics, update menu and manage employee shifts. The database contains 5 sub-packages namely – Users, Schedule, Menu, Inventory and Employee portal. The database contains information about the Menu and the Inventory, such as the Food categories, Price, different types of ingredients and their quantity. The employee portal contains data about the employee preferred shifts. The database also stores statistical results generated by the processor. The processor package contains 3 sub-packages namely – Cluster, Statistics and Dataset. The processor analyses the data retrieved from the database and generates statistical reports and suggestions. The controller manages the information flow involving the database and the other packages in the system.

14.3 Persistent Data Storage

For the proper working of the system, the data collected is to be stored in required formats. To create such a database, we use MYSQL in our project. This includes inventory management, Employee shift timings and menu updates. In SQL, all these are entered in terms of tables and fields which would help in the permanent storage of the data.

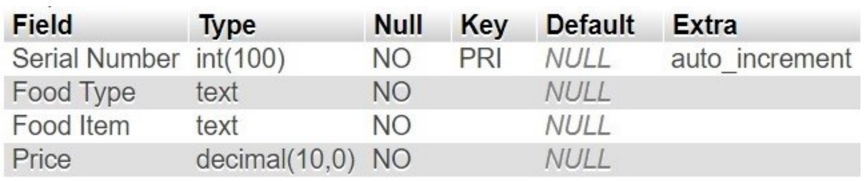
1. Inventory Management Database Schema:



Field	Type	Null	Key	Default	Extra
index	int(100)	NO	PRI	*NULL*	auto_increment
Food item	text	NO		*NULL*	
Ingredients	varchar(100)	NO		*NULL*	

This table stores the value of the food item and their corresponding ingredients. The ingredients table would have a list of all the ingredients required for a particular food item and the quantity of the ingredient remaining.

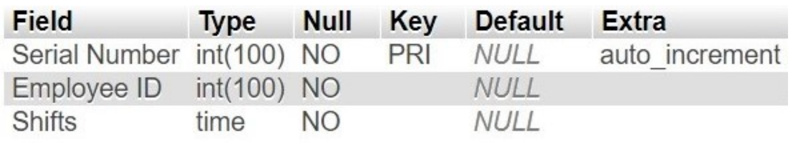
2.Menu Update Database Schema:



Field	Type	Null	Key	Default	Extra
Serial Number	int(100)	NO	PRI	*NULL*	auto_increment
Food Type	text	NO		*NULL*	
Food Item	text	NO		*NULL*	
Price	decimal(10,0)	NO		*NULL*	

This database has the fields for the different food items, food type and the price of each item. They are all inputted as separate fields in the table. The food type can be classified as appetizers, main course and deserts. All the food items will be grouped into these three categories.

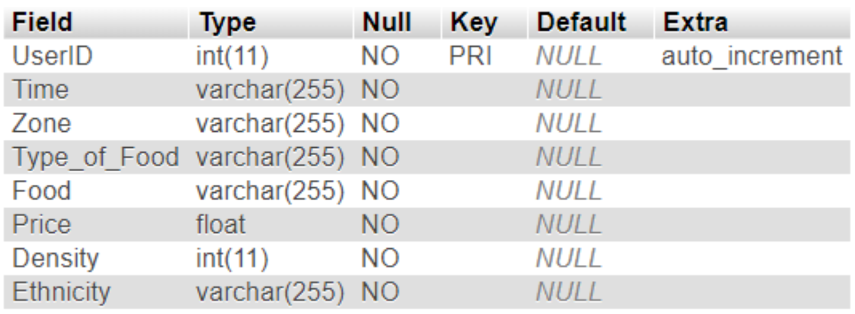
3.Employee Shift Database schema:



Field	Type	Null	Key	Default	Extra
Serial Number	int(100)	NO	PRI	*NULL*	auto_increment
Employee ID	int(100)	NO		*NULL*	
Shifts	time	NO		*NULL*	

This database contains the Shift details of the employees. Each employee can be identified with their employee ID and their preferred working shifts are inputted in the table and fields.

4.User Database schema:



Field	Type	Null	Key	Default	Extra
UserID	int(11)	NO	PRI	*NULL*	auto_increment
Time	varchar(255)	NO		*NULL*	
Zone	varchar(255)	NO		*NULL*	
Type_of_Food	varchar(255)	NO		*NULL*	
Food	varchar(255)	NO		*NULL*	
Price	float	NO		*NULL*	
Density	int(11)	NO		*NULL*	
Ethnicity	varchar(255)	NO		*NULL*	

This table with all the field entry is used as an artificial dataset for our project. It contains various fields such as location, customer density and the ethnicity. The analytics with different parameters are performed using this dataset.

14.4 Network Protocol

We will be using the XAMPP server which offers the ability to create a local web server for testing purposes. XAMPP is an open source free software developed by Apache friends. Its software package contains Apache distributions for Apache server, MariaDB, PHP, and Perl, and it is basically a local host or a local server. XAMPP also provides support for creating and manipulating databases in MariaDB and SQLite among others. We use the XAMPP server to generate our datasets, perform data analysis on these datasets, and then display the results of the analysis to the user.

14.5 Global Control Flow

Execution Order

The “Foodbytes” software has the following order of execution:

1. The order of the customer is received by the waiter and entered into the system.
2. This order is then stored in the database.
3. Depending on the order sent to the database, the inventory management system is subsequently updated with quantity of available items.
4. Along with step 3, the data analysis algorithms based on the different factors (customer density, revenue maximization, geographic location).
5. The manager is updated with alerts when items in the inventory meet the threshold and restocking is required.
6. The manager also receives suggestions and predictions obtained from the data analysis performed on the database so that he can be more prepared for future demands.

   The order of execution is initially sequential. That is, till step 3, each process depends on the previous, and therefore happens one after another. Step 3 and 4 are independent of each other and are hence carried out periodically. Step 5 is the result of step 3 and step 6 is the result of step 4. Hence both these steps are also independent of each other.

Time Dependency

The time dependency of the different processes varies in the “Foodbytes” software. The initial step of receiving the order depends on real time depending on the customer in flow in the enters an order to the system. The same goes for the inventory management system. The system depends on availability of items of the inventory and therefore is non-periodic. The data analysis is periodic, different analysis factors may have different time periods. This is dependent on the factors that they are analyzing and the requirements of analysis.

Concurrency

As mentioned above, the initial processes are not concurrent and happen one after another. The four data analysis processes, which are Analysis based on customer density, Trend analysis based on geographic location, Item based revenue maximization and Inventory management happen simultaneously. However the time of the results may vary as this will depend on the time the different algorithms take to execute. Different processes will have different sizes of data to process and this too, will result in different times of the result.

14.6 Hardware Requirements

The functionalities of the software "Foodbyte" are designed to work on a Windows Personal Computer. The minimum and recommended specifications of the Windows PC are listed below.

Table 22- Hardware requirements

Hardware Components	**Minimum Requirements**	**Recommended requirements**
Processor	1.8GHz dual core	2.0GHz dual core
RAM	2GB	4GB
Hard Drive Space	256GB	512GB
Network	WiFi 802.11 b/g/n	WiFi 802.11 a/ac/bg/n
Screen Size	15.6 inches	15.6 inches
Resolution	1280 x 800	1920 x 1080

15. ALGORITHMS AND DATA STRUCTURES

15.1 Inventory Management

In Inventory Management, we use a couple of algorithms to maximize the capacity of the system. One algorithm is used by the system to check the food vs the ingredients used whenever the customer orders some food. It will decrement the quantity of the ingredients used in the food product from the list that is available in the system.

When we pull the data from the database using php library in python, data comes with some overhead. An algorithm is used which uses string manipulation to reduce the overhead and the data is converted into a list of food items vs the ingredients used. Another list that we have is ingredients vs the quantity available.

Whenever the customer orders food using the POS (Point of Sale) device, the inventory management system uses the first table of food vs the ingredients used to find the ingredients that are used in preparing the food and updates the ingredient vs quantity list ingredient by ingredient.

15.2 Data Structures and Algorithms

If the behavior and rate of consumption of our food products in response to the handling conditions of the supply chain is understood, logistics can be improved.

We analyze the rate of consumption of each ingredient known as a perishable and generate timely quotes and suggest changes in the menu in order to avoid wastage of inventory.

The strategy we use is FEFO (First Expired First Out). FEFO makes different assumptions in terms of product shelf life, it will only order goods when the expiry date is known thus assuming only high quality products arrive in a restaurant.

1) Dataset creation

The dataset will be created based on the Gaussian or Normal probability distribution. Each module e.g. Location, Customer Density and Ethnicity, will create a set of table entries based on the initial observation data stored in a separate csv file. These table entries will be merged into a final set of entries by the probability distribution model based on the variance and mean of the input data. The input dataset will be divided into p subsets, where p will be calculate using normal distribution function. Finally, all p subsets will be written to the database.

Algorithm

Location module –

- Initialize hash-mapfor Food Category: FoodEntry{list}, Location:Zone
- Read the input observations in csv file usingPandas
- Calculate the percentage value of each observation in the dataframe
- Scale the percentage values based on the total of eachrow
- Add rounding correction based on the number of dataset entries
- Use the corrected percentage values to get the number of entries of each Food Category for a givenZone Value.
- Access the Food Category hash-map, Zone hash-map and the calculated number of entries of each type to create the {Zone, Food Entry}list

Ethnicity module –

- Initialize hash-map for Ethnicity: Zone
- Read the input observations in csv file using Pandas
- Calculate the percentage value of each observation in the data frame
- Scale the percentage values based on the total of each row
- Add rounding correction based on the number of dataset entries
- Use the corrected percentage values to get the number of entries of each Ethnicity for a given Zone Value.
- Access the Ethnicity hash-map, Zone hash-map and the calculated number of entries of each type to create the {Zone, Ethnicity} list

Customer Density module –

- Initialize hash-map for Time: FoodType
- Read the input observations in csv file using Pandas
- Calculate the percentage value of each observation in the data frame
- Scale the percentage values based on the total of each row
- Add rounding correction based on the number of dataset entries
- Use the corrected percentage values to get the number of entries of customers for a given Time Zone Value.
- Access the Food Type hash-map, Time Zone hash-map and the calculated number of entries of each type to create the {Time, Customer Density, Food Type}list

Merge module –

- Get the individual table entries for all modules
- Use the number of entries table of each module to find the number of entries in the final table based on probability distribution
- Access the Food Category, Ethnicity hash-map, Zone hash-map, Time hash-map, and the calculated number of entries of each type to create the {Zone, Food Category, Ethnicity, Time,customer_density} list
- Insert this table into the Users database

2) Clustering

The clustering algorithm will read the database and divide the entries based on the type of analysis. For clustering based on location (Zone), the entire table will be divided into n clusters, where n is the number of zones. These n clusters will then be subdivided into p clusters based on the Food Category. These p clusters will then be subdivided into q clusters based on Ethnicity or Time zone, based on the analysis requirement. The total number of clusters will be (n * p * q).

Pseudo Code:

```
Entry = ReadTable(Users, Database) // Read table from Database
n, p, q = sizeof(Zone), sizeof(Ethnicity), sizeof(Time) // Num of clusters
// Create n clusters based on Zone for all values in Entry:
for each zone:
if (value is equal to the Zone) copy row into ith cluster

// Create p clusters based on Food Category for all values in each Zone cluster:
for each Food_category:
if (value is equal to the Food_category) copy row into jth cluster

// Create q clusters based on Ethnicity
for all values in each Food Category cluster: for each ethnicity:
if (value is equal to the ethnicity) copy row into kth cluster
```

3) Statistical analysis

For each cluster in (n * p * q), we find the popularity of a food category based on the number of entries in that cluster. Based on the type of analysis, the Food categories will be placed in a list with decreasing order of popularity. For location and Ethnicity based analysis, for ith zone and jth ethnicity the Food categories with maximum will be calculated and stored in the database along with zone and ethnicity.

For customer density, the number of customers at a specific location will be clustered based on the time. From these clusters, we will store the maximum density times during the day in the database.

Pseudo code

```
// Calculate Popularity Index of each Food category
Popularity hash-map = {Parameter[]:popularity_index} for each cluster in (n * p * q):

For each Food category:
    count the number of entries of Food category in ith cluster insert
the count in the popularity hash-map in decreasing order
For each entry in Popularity list
    insert into database Parameter[i], Parameter[j], …., Popularity_index


// Display results
Data = Read popularity database
Plot graphs of popularity of Food category for varying values of Location, Ethnicity and Time Zone
on the Interface

Data1 = Read Employee database
Plot graph for number of current vs required employees over time t on the Interface
```

4) Suggestion based on analysis

Based on the analysis algorithms, the database for Food category popularity and customer density will be stored in separate tables. This algorithm will read the tables, find the maxima and minima and provide a list of suggestions based on the retrieved data.

Pseudo Code

```
Data1 = ReadTable(Popularity)
Data2 = ReadTable(TimeDensity)

// Menu update
for each entry in Data1
        search for the Food Category with the lowest and highest popularity Suggest price update for
the least popular Food category at each location
```

```
// Employee count

EmployeeCount = ReadTable(EmployeeCount)

EmployeeInfo = ReadTable(EmployeeInfo)

for each entry in Data2

    search for Time with highest and lowest customer density

    save entries to TimeDensity table

for each time value
    Get current employee count at time t
    Get required employee count from TimeDensity table
    if(current employee count is less than Required employee count)

    get list of employees with preferred shift in EmployeeInfo overlap with time t
    select a random employee from the available list

    current employee count + 1
    Write Table(New Employee Shift Info)
```

16. USER INTERFACE AND DESIGN:

The graphical user interface part of our project is mainly confined to the manager, since our project plan mainly on providing the results of statistics and other suggestions to the management. For our project the front end is done with HTML and CSS and the back end is done using the MYSQL database. We have the project done on XAMPP server which runs on the localhost. PHP is our server-side scripting language.

The manager page starts with a login. The manager would be able to access his profile using his login and password. In the home page, all the facilities which can be updated by the manager will be displayed. They will be displayed in the form of buttons. All the designing is done using HTML and CSS was used to style the HTML elements. From his main page the manager can choose options to update the menu, waiter shifts, manage waiter wages and view profit/loss statements.

Furthermore, in the analytics section the manager can view the different analysis performed with various parameters from which it would be useful for him to take efficient decision in the administration part. Although these pages like all other pages will be served dynamically, these pages themselves will be served from static HTML styled with CSS styling sheet content. When the manager enters into any of the analytics part the graph and the statistics will be displayed. The analytics part is still in process. These pages merely serve content to the end-user and no content needs to be sent from the client to the server, making these pages very simple to render.

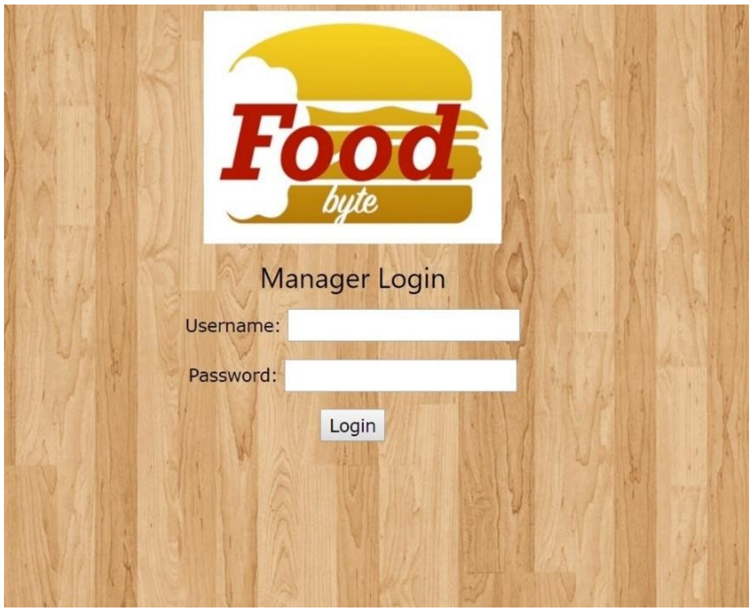
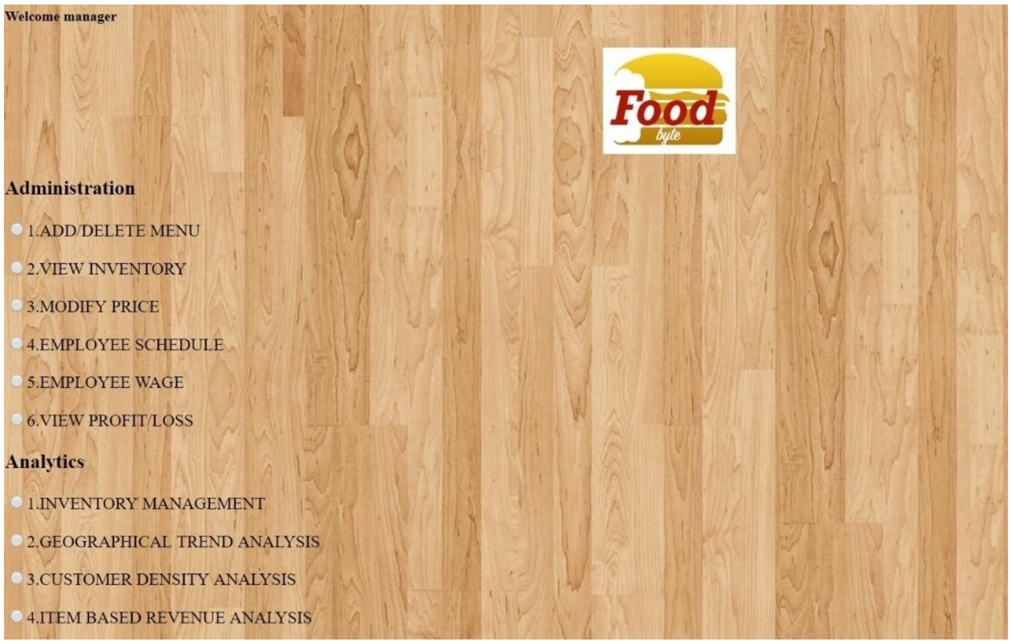


Figure 31&32: Manager before and after login page



Administration

- 1.ADD/DELETE MENU
- 2.VIEW INVENTORY
- 3.MODIFY PRICE
- 4.EMPLOYEE SCHEDULE
- 5.EMPLOYEE WAGE
- 6.VIEW PROFIT/LOSS

Analytics

- 1.INVENTORY MANAGEMENT
- 2.GEOGRAPHICAL TREND ANALYSIS
- 3.CUSTOMER DENSITY ANALYSIS
- 4.ITEM BASED REVENUE ANALYSIS

17. DESIGN OF TESTS:

Menu:

Test-Case Identifier: TC-01
Function Tested: addToMenu(string itemName, double Price): bool
Pass/Fail Criteria: Test will pass if a new Item is added to Menu List.

Test Procedure	Expected Results
Call Function (Pass) Call Function (Fail)	Item will be added to the menu. If similar name and price is already there, it will return false. If only price is different it will update Price and return true.

Inventory:

Test-Case Identifier: TC-02
Function Tested: UpdateIngredient(Ingredient, FoodEntry, Action):bool
Pass/Fail Criteria: Test will pass if an existing Item is removed from ingredient List.

Test Procedure	Expected Results
Call Function **(Pass) Call Function (Fail)**	**Ingredient will be removed from the ingredient list.** **Function will fail if the item does not exist in the ingredient list.**

Test-Case Identifier: TC-03
Function Tested: UpdateAvailableCount(Ingredient):int
Pass/Fail Criteria: The test will pass if thecount of Ingredient is Updated in the Database.

Test Procedure	Expected Results

Call Function (Pass) Call Function (Fail)	Ingridient count will be updated in the database. If updates are not available the function returns nothing and no changes are made to the database.

Test-Case Identifier: TC-04
Function Tested: CheckRefill(Ingredient,count):bool
Pass/Fail Criteria: The function will pass if ingredient count is below the recommended threshold.

Test Procedure	Expected Results
Call Function (Pass) Call Function (Fail)	Function will return true if the quantity of an ingredient is below its threshold. If no item‟s quantity is below the threshold value then the function will return false.

Test-Case Identifier: TC-05
Function Tested: GetRefillStatistics():list<Ingredient_list*>
Pass/Fail Criteria: The function will pass if itReturns a list of all the ingredients that needed refill.

Test Procedure	Expected Results
Call Function (Pass) Call Function (Fail)	Function will return list of ingredients that need refill. If no item requires a refill then the function will return false.

Test-Case Identifier: TC-06
Function Tested: GetExpiryList(Ingredient_list *): list<expiry_info*>
Pass/Fail Criteria: The function will pass if itReturns the expiry period of all the items in the Ingredient list.

Test Procedure	Expected Results
Call Function (Pass)	Function will return list of ingredient‟s expiry period. Function will not return the list.

Call Function(Fail)	

Schedule:

Test-Case Identifier: TC-07
Function Tested: GetShiftInfo(EmployeeID, shiftInfo *)
Pass/Fail Criteria: The function will pass if itReturns the shift information for the given employee ID.

Test Procedure	Expected Results
Call Function (Pass) Call Function (Fail)	Function will return the shift of a given employee. If the employee ID is not correct or the information does not exists in the database then the function will return false.

Test-Case Identifier: TC-08
Function Tested: ChangeShift(EmployeeID, shiftInfo)
Pass/Fail Criteria: The function will pass if itUpdates shift of the employee and sends it to database.

Test Procedure	Expected Results
Call Function (Pass) Call Function (Fail)	Function will return the updated shift of a given employee. If the employee ID is not correct or the information does not exists in the database then the function will return false.

Test-Case Identifier: TC-09
Function Tested: ChangeShift(EmployeeID, shiftInfo)
Pass/Fail Criteria: The function will pass if itUpdates shift of the employee and sends it to database.

Test Procedure	Expected Results

Call Function (Pass) Call Function (Fail)	Function will return the updated shift of a given employee. If the employee ID is not correct or the information does not exists in the database then the function will return false.

Employee Info

Test-Case Identifier: TC-10
Function Tested: AddEmployee(name,ID,Title,shiftInfo*):bool
Pass/Fail Criteria: The function will pass if itAdds new employee information into the database.

Test Procedure	Expected Results
Call Function (Pass) Call Function (Fail)	Function will add employee"s info to the database. If the employee info exists in the database then the function will return false.

Test-Case Identifier: TC-11
Function Tested: RemoveEmployee(ID):bool
Pass/Fail Criteria: The function will pass if itRemoves an employee from database.

Test Procedure	Expected Results
Call Function (Pass) Call Function (Fail)	Function will remove employee"s info from the database. If the employee info does not exist in the database then the function will return false.

Test-Case Identifier: TC-12
Function Tested: UpdateShift(ID,shiftInfo*):bool
Pass/Fail Criteria: The function will pass if itUpdates the preferred shift information for the employee.

Test Procedure	Expected Results

Call Function (Pass) Call Function (Fail)	Function will update employee‟s info to the database. If the employee‟s preferred shift is not availablebthen the function will return false.

Cluster:

Test-Case Identifier: TC-13
Function Tested: Geography(DataInfo *): int
Pass/Fail Criteria: The function will pass if itReturns the cluster index of the DataInfo set based on FoodTypevs Location clustering.

Test Procedure	Expected Results
Call Function (Pass) Call Function (Fail)	Function will return the type of cluster the data belongs to. If the data does not fall under geography category function will fail.

Test-Case Identifier: TC-14
Function Tested: customerdensity(DataInfo *): int
Pass/Fail Criteria: The function will pass if itReturns the cluster index of the DataInfo set based on time vs customer density clustering.

Test Procedure	Expected Results
Call Function (Pass) Call Function (Fail)	Function will return the type of cluster the data belongs to. If the data does not fall under customer density category function will fail.

Test-Case Identifier: TC-15
Function Tested: itemrevenue(DataInfo *): int
Pass/Fail Criteria: The function will pass if itReturns the cluster index of the DataInfo set based on time vsFoodType clustering.

Test Procedure	Expected Results

Call Function (Pass) Call Function (Fail)	Function will return the type of cluster the data belongs to. If the data does not fall under customer item revenue category function will fail.

Statistics

Test-Case Identifier: TC-16
Function Tested: ShiftAnalysis(clusterInfo *,shiftInfo *,predictInfo *):bool
Pass/Fail Criteria: The function will pass if itReturns true if the analysis predicts any updates.

Test Procedure	Expected Results
Call Function (Pass) Call Function (Fail)	Performs shift analysis on the input cluster and updates the prediction result based on the preferred shift of each employee. Returns true if the analysis predicts any updates. If there is no analysis it will return false.

Test-Case Identifier: TC-17
Function Tested: RevenueAnalysis(clusterInfo *, predictInfo *):bool
Pass/Fail Criteria: The function will pass if itReturns true if there is price variation over a period of time.

Test Procedure	Expected Results

Call Function (Pass) Call Function (Fail)	Use the cluserInfo to check for price variation over a period of time and update the prediction result for price updates if needed. If there is no update in price it will return false.

Test-Case Identifier: TC-18
Function Tested: UpdateLists(Listtype, clusterInfo *):void
Pass/Fail Criteria: The function will pass if itReturns true if there areupdates in the menu list, price list or shift list in the input cluster.

Test Procedure	Expected Results
Call Function (Pass) Call Function (Fail)	Based on the listtype, updates the menu list, price list or shift list in the input cluster. If there is no update it will return false.

Manager Interface:

Test-Case Identifier: TC-19
Function Tested: View(InfoType *):void
Pass/Fail Criteria: The function will pass if itShows the statistical data based on the selection type from the interface.

Test Procedure	Expected Results

Call Function (Pass) Call Function (Fail)	Based on the interface selection, it shows the statistical data. If there is no update it will fail.

Test-Case Identifier: TC-20
Function Tested: removeFromMenu(iteminfo *):bool
Pass/Fail Criteria: The function will pass if itRemoves an item from the Menu database.

Test Procedure	Expected Results
Call Function (Pass) Call Function (Fail)	The function will return true if an entry is removed from the database. If the item is not in database it will return false.

Test-Case Identifier: TC-21
Function Tested: updatePrice(itemInfo *,NewPrice):bool
Pass/Fail Criteria: Updates price in Menu database.

Test Procedure	Expected Results
Call Function (Pass) Call Function (Fail)	The function will return true if the price of an item is updated in database. If the item is not in database it will return false.

Controller:

Test-Case Identifier: TC-22
Function Tested: ConnectDB(ID):bool
Pass/Fail Criteria: The test passes if the controller is able to connect to and access the database.

Test Procedure	Expected Results
Call function (Pass)	Controller is able to connect to the database
Call function (Fail)	If connection cannot be established, function will return false.

Test-Case Identifier: TC-23
Function Tested: RequestEntry(requestInfo *):void
Pass/Fail Criteria: The test passes if the controller receives valid data.

Test Procedure	Expected Results
Call function (Pass)	Controller is able to receive data from the database when the controller is connected to the database.
Call function (Fail)	The function will fail if the controller is not able to acess the database

Test-Case Identifier: TC-24
Function Tested: UpdateEntry(updateInfo *):void
Pass/Fail Criteria: The test passes if a the controller is able to update data in interface and database

Test Procedure	Expected Results
Call function (Pass)	Controller is able to update information in the database and interfaces.
Call function (Fail)	The function will fail if the controller is not able to update the data in database.

Test-Case Identifier: TC-25
Function Tested: DataAnalyze(): void
Pass/Fail Criteria: The test passes if the controller is able to run algorithms on data and store results in database.

Test Procedure	Expected Results
Call function (Pass).	Controller is able to run needed algorithms and updates information in database.
Call function (Fail)	If analysis fails, function fails.

Test-Case Identifier: TC-26
Function Tested: InterfaceConnection(): void
Pass/Fail Criteria: The test passes if the controller is able to connect to and access interfaces.

Test Procedure	Expected Results
Call function (Pass).	Controller is able to connect to the interfaces.
Call function (Fail)	If connection cannot be established, function fails.

Integration Testing Strategy

We deemed our system to be best suited for bottom up testing. This approach is more practical due to the structure of the modules in the application. For instance, if we consider the possibility that there could be a problem with the interaction between the food byte system and manager interface, i.e. results not displaying correctly. If we follow the bottom up strategy, we can test first the lowest level module i.e. the prediction module. Through which we can observe whether the module is functioning properly or not. If it is functioning properly then we can test the subsequent higher level modules. If we take a different approach to testing, it would be much more difficult to understand whether that issue comes down to the actual interaction or integration of the modules or if it is a problem with how the individual classes were originally designed.

For example, if an item requires refilling and the inventory results do not reflect this then by following the bottom up strategy, we can determine whether the problem lies in the implementation or the integration. Bottom up testing alleviates this quite a bit by first solidifying the foundation and moving onto the top of the system.

18. HISTORY OF WORK

18.1 Progress table

Task Description	Start date	End date
Full Project	9/23/18	12/11/18
Report 1	9/19/18	10/7/20
Dataset Creation	10/2/18	10/25/18
Report 2	10/21/18	11/11/18
Interface Diagrams	9/25/18	9/30/18
System Architecture	10/2/18	10/15/18
User Interface	10/2/18	11/10/18
Data Analytics Algorithms	10/10/18	11/15/18
Module Testing	11/16/18	11/20/18
Merge Individual Models	11/23/18	11/30/18
Integration Testing	11/31/18	12/9/18
Demo 1	10/31/18	10/31/18
Report 3	11/25/18	12/9/18
Demo 2	12/12/18	12/12/18

Table 23: Progress Table

First the dataset was created, and the next task was to analyze the data and use suitable machine learning algorithms for the various functionalities of the system.

For trend analysis based on geography, the relevant data from the database was be clustered based on location. Next, the data was be sub clustered based on ethnicity. The counter sale of each food category was calculated, and thresholds were defined for categorizing the results and providing suggestions. Finally, analytics graph was generated along with suggestions for price and menu updates.

For trend based on customer density, the database was read and clustered based on time and customer density. Based on customer density, the count of required employees at particular intervals were suggested. This compared with existing employee count, thresholds were identified. An increase or decrease in employee count at a particular time interval was suggested ensuring that the threshold is maintained. Statistics based on employee wage expenditure was maintained and a graph generated. Also, various offers on food items were suggested based on a particular time period and customer density.

For analyzing sales trends based on time, the database will be clustered based on time and item sales. Based on the past sales trend of each item based on time, the future sales trend will be predicted for a certain interval. Based on this, alteration of menu (items and food prices) will be suggested.

For the inventory management module, an ingredient lookup database was created. The inventory database was linked to menu the sales database and the count of inventory was reduced for the sale of every item. The data was refreshed every 16 days and the maximum and minimum consumed items per day for those 16 days were determined and plotted on a graph. The alerts section also displayed the items which were less than 30% of the full stock amount.

Once all the modules were completed, they were tested independently. Once independent testing was complete, they were merged, and a final integration was performed.

Project Management:

The team mainly communicates through 'Whatsapp' and meets at least once a week to discuss the tasks that have been completed and the current items in hand. During these meetings, we also split the responsibilities to be carried out throughout the week. Some amount of knowledge transfer also takes place during these gatherings as and when requested by a particular team member.

The group has a common media drive for uploading and sharing reports and other documents amongst the team. We also have a GitHub repository for code management.

In a scenario where a particular team member is unable to complete their assigned tasks due to unavoidable circumstances such as illness, that task is taken up by other members of the team.

In case of any issues or conflicts within the team, it is discussed and resolved amicably.

18.2 Product Ownership

We had divided our group of 8 team members into 4 teams with 2 members each. Each team worked on an independent data analytics module which was tested independently. We then integrated all the modules before integration testing. The teams are as follows: Yashasvi and Akshay, Prince and Sarthak, Anvitha and Priyanka, Suraj and Ashwin.

- Yashasvi and Akshay

Completed tasks: Created a first draft of the UI for customer login. Selected Gaussian distribution model for creating the artificial dataset. Identified key data points from online sources related to Food vs Geography distribution. Created the dataset based on the mathematical model. Identied the classifier needed to find different clusters based on this dataset.

- Suraj and Ashwin

Completed tasks: Database analysis for the required fields and entries needed for Item based revenue maximization. Worked on libraries and functions in Python that can be used for implementation. Worked on already created dataset and used data from specific columns in the Users Database for sales prediction.

- Anvitha and Priyanka

Completed tasks: Created a first draft UI of the Manager's account. Studied data analytics algorithms to determine how many employees are required each shift based on the number of customers. Analyzed the trend in customer density for the past months.

- **Prince and Sarthak**

Completed tasks: Created a first draft of the UI for chef and created the Inventory database for food item vs ingredient mapping. Linked the Ingredient lookup tables and Ingredient availability to the server-side program which created the required dataset. Used the generated dataset to find the maximum and least consumed items per day.

19. CYCLOMETRIC COMPLEXITY CALCULATION:

1) Dataset creation

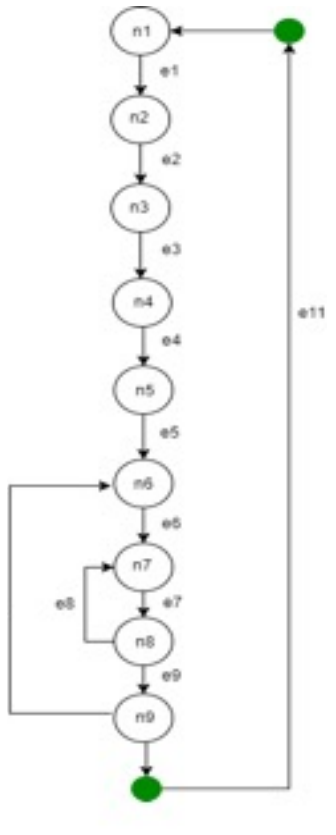


Figure 33: Complexity diagram for dataset creation

Same graph for modules – Location, Customer Density and Ethnicity Cyclometric complexity is

V(G) = E (edges) – N (nodes) + 2 * p V(G) = 11 – 9 + 2

V(G) = 4

2) Clustering

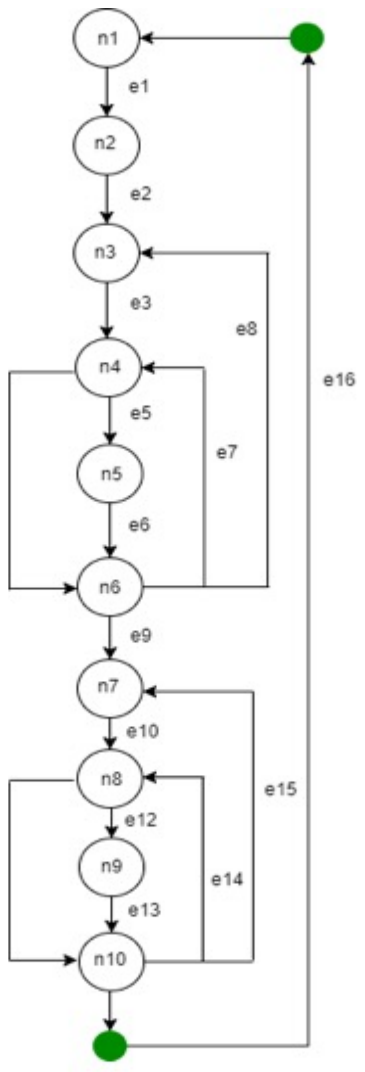


Figure 34: Complexity diagram for clustering

Graph for clustering based on Food category(1stloop) and sub-clustering based on Ethnicity(2ndloop)

Cyclometric complexity is

V(G) = E (Edges) – N (Nodes) + 2 * p V(G) = 16 – 10 + 2

V(G) = 8

3) Analysis

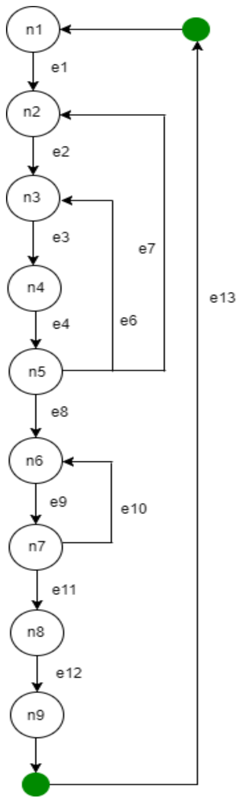


Figure 35: Complexity diagram for analysis

Graph for statistical analysis modules Cyclometric complexity is

V(G) = E(Edges) – N(Nodes) + 2 * p V(G) = 13 – 9 + 2

V(G) = 6

20. REFERENCES

1. Database managementhttps://www.apachefriends.org/index.html
2. Creating Menu Database – Panera Bread

https://delivery.panerabread.com/menu/category/?gclid=EAIaIQobChMI95zS-MjU3QIV0gOGCh1pmgpoEAAYASAAEgLa3fD_BwE&gclid=EAIaIQobChMI95zS-MjU3QIV0gOGCh1pmgpoEAAYASAAEgLa3fD_BwE

3. Creating datasets –Kaggle

a) https://www.kaggle.com/uciml/restaurant-data-with-consumer-ratings
b) https://www.kaggle.com/ulabox/ulabox-orders-with-categories-partials-2017
c) https://www.kaggle.com/nypl/whats-on-the-menu/version/1

4. Reports

http://www.ece.rutgers.edu/~marsic/books/SE/projects/Restaurant/2015-g3-report3.pdfhttp://www.ece.rutgers.edu/~marsic/books/SE/projects/Restaurant/2014-g4-report3.pdf

5. Books

http://eceweb1.rutgers.edu/~marsic/books/SE/book-SE_marsic.pdf

6. UML Designhttps://www.draw.io/
7. Data references

https://www.webstaurantstore.com/article/138/restaurant-inventory-management.htmlhttps://www.more.com/east-and-west-coast-cuisine-whats-difference